INTRODUÇÃO À PROGRAMAÇÃO

500 Algoritmos Resolvidos

Anita Lopes
Guto Garcia

15ª Tiragem

ELSEVIER

CAMPUS

Capa
RioTexto

Copidesque
Adriana Kramer

Editoração Eletrônica
RioTexto

Revisão Gráfica
Ivone Teixeira

Estratégia competitiva

ISBN 13: 978-85-352-1019-4
ISBN 10: 85-352-1019-9

Nota: Muito zelo e técnica foram empregados na edição desta obra. No entanto, podem ocorrer erros de digitação, impressão ou dúvida conceitual. Em qualquer das hipóteses, solicitamos a comunicação ao nosso Serviço de Atendimento ao Cliente, para que possamos esclarecer ou encaminhar a questão.

Nem a editora nem o autor assumem qualquer responsabilidade por eventuais danos ou perdas a pessoas ou bens, originados do uso desta publicação.

CIP-Brasil. Catalogação-na-fonte.
Sindicato Nacional dos Editores de Livros, RJ

L85i

Lopes, Anita
Introdução à programação / Anita Lopes, Guto Garcia. – Rio de Janeiro : Elsevier, 2002 – 15ª Reimpressão.

Apêndices
ISBN 85-352-1019-9

1. Programação (computadores). 2. Algoritmos. I. Garcia, Guto. II. Título

02-0186 CDD – 005.3
CDU – 004.42

Aos meus pais, Augusto e Maria Luíza, pela vida.
Ao meu filho, João Renato, pela vinda.
À Professora Sandra Mariano pela credibilidade do meu trabalho.

Anita Lopes

A todos os professores de algoritmos pelo desafio que
é a difícil missão de introduzir um novo conhecimento.

Aos alunos pelo esforço que farão para adquirir este
conhecimento que é a base do curso de Informática.

Guto Garcia

Agradecimentos

A Deus, pela concretização de um desejo.

Ao meu filho, por entender todos os momentos que não pude estar com ele para poder elaborar este livro.

Às minhas irmãs, cunhados e sobrinhos, pelo carinho e força que me deram.

Às professoras da UNESA-Bispo (Gellars Tavares, Jane Silva, Letícia Winkler, Mai-Ly Vanessa e Sonia Moreira), pela colaboração em vários enunciados.

Aos meus ex-alunos da UNESA, pela contribuição em algumas soluções deste livro.

ANITA LOPES

À minha mãe e ao meu pai, pela sabedoria, inteligência e amor.

Aos meus irmãos, pelas alegrias e cotidiano sempre divididos.

Agradeço a Valmir C. Barbosa, Sandra Cardozo de Abreu, José Luís Coelho Marques, Juarez Muylaert Filho, Sandra Mariano e Hilana Erlich que, de formas diferentes, me incentivam e apóiam na construção do meu caminho no mundo acadêmico.

Agradeço aos inúmeros ex-alunos e monitores do curso de informática da PUC-Rio e da Estácio de Sá, que, nos últimos anos, contribuíram muito na solução de exercícios.

GUTO GARCIA

Prefácio

Em informática, o ensino de algoritmos imperativos é um dos mais complicados. Os meus colegas (e eu mesmo!) não se cansam de discutir novas estratégias para introduzir conceitos como o de atribuição e de vetores. O problema principal, pelo menos no meu entender, é que nossos pupilos foram acostumados, no ensino fundamental e médio, a utilizar a Matemática como "modelo computacional", que é distante do modelo imperativo, a máquina de Von Neumann, que a eles é apresentada, na esmagadora maioria das vezes, em todos os cursos universitários introdutórios de programação, sejam eles em universidades brasileiras ou estrangeiras.

Contudo, estratégias, matemáticas e máquinas à parte, há uma certeza no ensino de algoritmos: só aprende quem pratica. Todos os meus colegas são unânimes em dizer aos alunos que não há muito mistério em entender o que um algoritmo faz mas que a questão fundamental reside na capacidade de construí-lo. Portanto, há que se praticar exaustivamente.

Neste sentido, Anita e Guto, juntando suas vastas expetiências acadêmicas no delicado ensino de algoritmos, realizaram um valioso trabalho em *Introdução à Programação – 500 Algoritmos Resolvidos*. Certamente, o leitor ou leitora, aprendiz de programação, mas empreendedor que, com afinco, implementar os algoritmos antes de olhar as soluções fornecidas, obterá uma sólida formação nas estruturas de controle, repetição, armazenamento e outras, fundamentais em toda e qualquer linguagem imperativa.

Juarez Assumpção Muylaert Filho
Diretor do Curso de Informática da UNESA
Professor Adjunto da UERJ

Sumário

Apresentação

Quando pensamos em produzir um livro sobre algoritmos, tínhamos a clareza de que ele deveria ter um diferencial; afinal, já existiam muitos no mercado.

Partimos do princípio de que ele deveria ter uma teoria e, principalmente, 500 algoritmos resolvidos, pois esta era a maior solicitação de nossos alunos: as soluções dos exercícios que passávamos.

Além disso, nossa maior preocupação era com a linguagem, que deveria ser a de uma pessoa leiga no assunto. Reunimos nossos materiais, anotamos dúvidas que surgiam durante as aulas e começamos nosso trabalho, que, aliás, teve muita participação de nossos alunos.

Ao longo desses quase três anos, fizemos muitas pesquisas e inventamos muitos enunciados para chegar ao número estipulado por nós.

Decidimos que não iríamos além do assunto funções para que pudéssemos explorar ao máximo os assuntos abordados no livro.

Testamos todos os algoritmos até vetores no interpretador sugerido por nós, UAL e, todos que foram possíveis, no ILA.

Hoje, ao vermos nosso trabalho impresso, temos certeza de que era isso que gostaríamos de fazer e esperamos poder ajudar a você, leigo ou não.

Sabemos que um livro jamais substituirá a figura do professor, mas esperamos que o nosso faça você se sentir bem perto de nós, porque esse foi o nosso maior investimento.

ANITA LOPES & GUTO GARCIA

Introdução

O aprendizado de Algoritmos nos cursos de graduação de Informática, Engenharia e Matemática, de acordo com da nossa experiência, é um processo extremamente difícil, tendo em vista o grande número de informações que os alunos precisam absorver em pouco tempo. Desta forma, temos observado uma desistência significativa da disciplina logo após a primeira avaliação.

As opiniões quanto ao ensino de algoritmos divergem muito, pois alguns professores defendem a tese de abstração, isto é, o aluno desenvolve no papel e não tem nenhum contato com a máquina.

Acreditamos que o aprendizado só se torna possível após a passagem do concreto para o abstrato; em outras palavras, o aluno precisa visualizar o que está fazendo e para isso o uso de um interpretador em português irá ajudá-lo nessa etapa inicial.

Nosso livro é resultado de um trabalho que vem sendo elaborado há algum tempo com nossos alunos. Nesses últimos anos, começamos a fazer todas as soluções dos exercícios passados para eles e percebemos que o aprendizado melhorou significativamente, uma vez que se torna inviável a resolução de um grande número de exercícios durante as aulas. Além disso, os alunos ficam muito inseguros nesse primeiro contato com a programação, pois é um processo muito solitário.

Este livro é voltado para o ensino inicial de algoritmos e procuramos fazer isso de uma maneira bem simples: o livro possui conceitos teóricos sobre algoritmos de uma forma bem rápida e resumida e 500 exercícios resolvidos, uma vez que acreditamos que o iniciante em programação precisa praticar muito para poder depois abstrair.

A sintaxe que usamos está mais parecida com a linguagem C e sugerimos o uso de um interpretador que roda sob ambiente Linux e cujas informações encontram-se no Apêndice I deste livro.

Muitas perguntas feitas por nossos alunos foram incorporadas a este livro porque podem ser também as suas dúvidas. Além disso, para explicar melhor o acompanhamento da execução do algoritmo, apresentaremos a saída no vídeo e a alocação da Memória Principal em relação às variáveis usadas para que você possa ir conhecendo um pouco mais sobre esse processo.

Os 500 algoritmos resolvidos estão divididos por assunto e organizados em 6 grandes blocos: o primeiro abrange algoritmos do cotidiano; no segundo, utilizamos somente algoritmos que usam funções, comandos de atribuição, de entrada e saída; no terceiro bloco, veremos o comando de seleção; no quarto bloco, os comandos de repetição; no quinto, utilizamos os algoritmos que manipulam vetores e matrizes, e no sexto, agrupamos algoritmos utilizando funções. Nos apêndices, falamos sobre uma ferramenta para testar os algoritmos no computador, código ASCII e apresentamos problemas de raciocínio lógico.

Não pule etapas. Procure fazer todos os exercícios de um capítulo antes de passar para outro. Estaremos com você em cada página, em cada comentário e temos certeza de que você aproveitará muito.

O aprendizado de algoritmos é algo apaixonante desde que você acredite e invista.

Qualquer dúvida disponibilizamos nossos e-mails:

Anita Lopes: anitalml@iis.com.br

Guto Garcia: guto@cos.ufrj.br

OS AUTORES

Capítulo I

Conceitos iniciais

■ **Lógica de programação** é a técnica de encadear pensamentos para atingir determinado objetivo. O aprendizado desta técnica é necessário, para quem deseja trabalhar com desenvolvimento de sistemas e programas.

■ **Algoritmo** é uma seqüência de passos finitos com o objetivo de solucionar um problema.

Quando nós temos um problema, nosso objetivo é solucioná-lo.

Algoritmo não é a solução de um problema, pois, se assim fosse, cada problema teria um único algoritmo. Algoritmo é um conjunto de passos (ações) que levam à solução de um determinado problema, ou então, é um caminho para a solução de um problema e, em geral, os caminhos que levam a uma solução são muitos.

O aprendizado de algoritmos não é uma tarefa muito fácil, só se consegue através de muitos exercícios. Este é o objetivo principal deste livro: possibilitar que você, a partir das soluções apresentadas, venha construir sua própria lógica de programação.

Todos nós, no dia-a-dia, nos deparamos com vários problemas. Quantas vezes já vimos um algoritmo e não sabíamos que aquela seqüência de passos chamava-se algoritmo. Um exemplo bem freqüente é quando queremos falar em algum telefone público. Aquilo que está escrito no telefone é um algoritmo. Veja a seguir um exemplo de um algoritmo do cotidiano.

```
1 – Retirar o telefone do gancho
2 – Esperar o sinal
3 – Colocar o cartão
4 – Discar o número
5 – Falar no telefone
6 – Colocar o telefone no gancho
```

O algoritmo é exatamente esse conjunto de passos que resolveu o problema de uma pessoa falar no telefone. É como se fôssemos ensinar uma máquina a fazer alguma tarefa específica.

Outro exemplo clássico é um algoritmo para resolver o problema de fritar um ovo que poderia estar escrito em forma de uma receita. A receita é um algoritmo, pois é formada de ações que devem ser tomadas para fritar o ovo. Como fritar um ovo é muito fácil, esta deve ser a primeira receita de fritar um ovo que vocês já leram.

algoritmo 1

```
1 – pegar frigideira, ovo, óleo e sal
2 – colocar óleo na frigideira
3 – acender o fogo
4 – colocar a frigideira no fogo
5 – esperar o óleo esquentar
6 – colocar o ovo
7 – retirar quando pronto
```

Cada linha do algoritmo podemos chamar de uma instrução, logo, podemos dizer que um algoritmo é um conjunto de instruções.

■ **Instrução** indica a um computador uma ação elementar a ser executada.

Até as coisas mais simples podem ser descritas por um algoritmo. Por exemplo: "Mascar um chiclete".

algoritmo 2

```
1 – pegar o chiclete
2 – retirar o papel
3 – mastigar
4 – jogar o papel no lixo
```

Outros exemplos:

algoritmo 3

Algoritmo para trocar lâmpadas

```
1 – se (lâmpada estiver fora de alcance)
     pegar a escada;
2 – pegar a lâmpada;
3 – se (lâmpada estiver quente)
     pegar pano;
4 – Tirar lâmpada queimada;
5 – Colocar lâmpada boa;
```

algoritmo 4

Algoritmo para o fim de semana

```
1 – vejo a previsão do tempo;
2 – se (fizer sol)
     vou à praia;
    senão
    vou estudar;
3 – almoçar
4 – ver televisão
5 – dormir
```

algoritmo 5

Algoritmo para fazer um bolo simples

```
1 – pegar os ingredientes;
2 – se (roupa branca)
     colocar avental;
3 – se (tiver batedeira)
      bater os ingredientes na batedeira;
      senão
     bater os ingredientes à mão;
4 – colocar a massa na forma;
5 – colocar a forma no forno;
6 – aguardar o tempo necessário;
7 – retirar o bolo;
```

algoritmo 6

Algoritmo para descascar batatas

```
1 – pegar faca, bacia e batatas;
2 – colocar água na bacia;
3 – enquanto (houver batatas)
     descascar batatas;
```

algoritmo 7

Algoritmo para fazer uma prova

```
1 – ler a prova;
2 – pegar a caneta;
3 – enquanto ((houver questão em branco) e (tempo não terminou))faça
     se (souber a questão)
      resolvê-la;
     senão
      pular para outra;
4 – entregar a prova;
```

algoritmo 8

Algoritmo para jogar o jogo da forca:

```
1 – escolher a palavra;
2 – montar o diagrama do jogo;
3 – enquanto ((houver lacunas vazias) e (corpo incompleto))faça
    se (acertar uma letra)
     escrever na lacuna correspondente;
    senão
     desenhar uma parte do corpo na força;
```

algoritmo 9

Algoritmo para jogar o jogo da velha:

```
enquanto ((existir um quadrado livre ) e (ninguém perdeu(ganhou) o jogo))
     espere a jogada do adversário, continue depois
     se (existir um quadrado livre)
      se (centro livre)
        jogue no centro
      senão
        se (adversário tem 2 quadrados em linha com o terceiro
            desocupado)
            jogue neste quadrado desocupado
        senão
         se (há algum canto livre)
           jogue neste canto
```

algoritmo 10

Algoritmo para levar um leão, uma cabra e um pedaço de grama de um lado para outro de um rio, atravessando com um bote. Sabe-se que nunca o leão pode ficar sozinho com a cabra e nem a cabra sozinha com a grama.

```
1 – Levar a grama e o leão
2 – Voltar com o leão
```

```
3 – Deixar o leão
4 – Levar a cabra
5 – Deixar a cabra
6 – Voltar com a grama
7 – Levar o leão e a grama
```

DESAFIO

1) Fazer um algoritmo para levar 3 missionários e 3 canibais de um lado para outro de um rio, atravessando com um bote. Sabe-se que nunca pode ter mais missionários do que canibais porque senão os missionários catequizam os canibais. O que fazer para levar os 6 de uma margem para outra?

Ao formularmos um algoritmo, temos de ter clareza a respeito do aspecto estático dessa seqüência de ações escritas num pedaço de papel. O aspecto dinâmico só se evidencia ao executarmos essa seqüência no computador e daí, poderemos ter certeza se conseguimos ou não resolver o problema.

Programa de Computador nada mais é do que um algoritmo escrito numa linguagem de computador (C, Pascal, Fortran, Delphi, Cobol e outras). É a tradução para o inglês do algoritmo feito em português. O mais importante de um programa é a sua lógica, o raciocínio utilizado para resolver o problema, que é exatamente o algoritmo.

Capítulo 2

Variável, expressões, funções, atribuição, entrada e saída

VARIÁVEL

Conceitos iniciais

Uma *variável* é um local na *memória principal*, isto é, um endereço que armazena um conteúdo. Em linguagens de alto nível, nos é permitido dar nome a esse endereço para facilitar a programação.

O conteúdo de uma variável pode ser de vários tipos: inteiro, real, caractere, lógico, entre outros. Normalmente, quando se ensina algoritmo, trabalha-se com os quatro tipos citados.

Uma vez definidos o nome e o tipo de uma variável, não podemos alterá-los no decorrer de um algoritmo. Por outro lado, o conteúdo da variável é um objeto de constante modificação no decorrer do programa, de acordo com o fluxo de execução do mesmo.

Em algoritmos, as variáveis serão definidas no início, por meio do comando definido:

tipo da variável	nome da variável	;

Os tipos que usaremos serão:

```
int a;
real b;
string nome;
logico r;
```

OBSERVAÇÃO

Quando formos dar nome às variáveis, se faz necessário seguirmos algumas regras. É bom ressaltar que estas regras irão variar de acordo com a linguagem. As seguintes regras:

1. *O primeiro caractere é uma **letra**.*
2. *Se houver mais de um caractere, só poderemos usar: **letra** ou **algarismo**.*
3. *Nomes de variáveis escritas com letras maiúsculas serão diferentes de letras minúsculas. Lembre-se: media é **diferente** de MEDIA.*
4. *Nenhuma palavra reservada poderá ser nome de uma variável.*

Nomes Válidos	***Nomes Não-Válidos***
media, alt, a2, PESO	*2w -> começa por algarismo* *media_aluno -> o caractere sublinha não é permitido* *peso do aluno -> o caractere espaço não é permitido*

Um dos objetivos de se declarar uma variável no início do algoritmo é para que seja alocada (reservada) uma área na memória (endereço de memória) para a variável. Outro objetivo da declaração de variáveis é que, após a declaração, o algoritmo sabe os tipos de operações que cada variável pode realizar; explicando: algumas operações só podem ser realizadas com variáveis do tipo inteiro, outras só podem ser realizadas com variáveis dos tipos inteiro ou real, outras só com variáveis de caractere etc.

Tipos de variáveis

Numérica

Variáveis numéricas são aquelas que armazenam dados numéricos, podendo ser divididas em duas classes:

int

Os números inteiros são aqueles que **não** possuem componentes decimais ou fracionários, podendo ser positivos ou negativos.

As variáveis compostas com esses números são chamadas de **VARIÁVEIS INTEIRAS.** Os elementos pertencentes aos conjuntos N e Z, apesar de serem

representáveis na classe dos números reais, são classificados como variáveis numéricas inteiras por não possuírem parte fracionária.

Como exemplo de números inteiros temos:

12	número inteiro positivo
-12	número inteiro negativo

---

*Normalmente, uma variável do tipo inteira poderá ocupar **1**, **2** ou **4 bytes** na **MP**.*

---

real

Os números reais são aqueles que podem possuir componentes decimais ou fracionários, podendo também ser positivos ou negativos.

As variáveis compostas com estes números pertencentes aos conjuntos dos números reais são chamadas de **VARIÁVEIS REAIS.**

Como exemplos de números reais temos:

24.01	número real positivo com duas casas decimais
144.	número real positivo com zero casa decimal
-13.3	número real negativo com uma casa decimal
0.0	número real com uma casa decimal

---

*Normalmente, uma variável do tipo real poderá ocupar **4** ou **8 bytes** na **MP**.*

---

string

Também conhecida como caractere, alfanumérica ou literal. Esse tipo de variável armazena dados que contêm letras, dígitos e/ou símbolos especiais.

Como exemplos de constantes string temos:

"Maria"	string de comprimento 5
"123"	string de comprimento 3
"0"	string de comprimento 1
"A"	string de comprimento 1

O número de **bytes** possíveis para armazenamento de uma variável **string** dependerá da linguagem, mas o mais importante é entender que uma variável string é armazenada na **MP** como sendo uma **matriz linha.** Observe o trecho de algoritmo a seguir e suponha que na entrada de dados foi digitado: ALGORITMOS.

string palavra;
leia palavra;

Em algumas linguagens, a numeração poderá começar com 1.

OBSERVAÇÃO

*Não confundir: **caractere que se encontra na posição 3** com **3º caractere**:*

No exemplo acima, temos:

- ***caractere que se encontra na posição 3** : **O***
- ***3º caractere** : **G***

logico

Também conhecido como booleano. É representado no algoritmo pelo dois únicos valores lógicos possíveis: verdadeiro ou falso. Porém, é comum encontrar em outras referências outros tipos de pares de valores lógicos como: sim/nao, 1/0, true/false, verdadeiro/falso.

Como exemplos de constantes lógicas temos:

verdadeiro	Valor lógico verdadeiro
falso	Valor lógico falso

OBSERVAÇÃO

As variáveis quando são declaradas, dependendo da linguagem, não têm nenhum valor atribuído; portanto, no início, atribua valores a todas as variáveis.

EXPRESSÕES

O conceito de expressão em termos computacionais está intimamente ligado ao conceito de expressão (ou fórmula) matemática, onde um conjunto de variáveis e constantes numéricas relaciona-se por meio de operadores compondo uma fórmula que, uma vez avaliada, resulta num valor. As expressões dividem-se em:

Aritméticas

Expressões aritméticas são aquelas cujo resultado da avaliação é do tipo numérico, seja ele inteiro ou real. Somente o uso de operadores aritméticos e variáveis numéricas é permitido em expressões deste tipo.

Como exemplo de operadores e expressões aritméticas temos:

Soma Na matemática, representada pelo sinal + e, em expressões em termos computacionais, pelo mesmo sinal.

A + B Expressão que simboliza a soma do valor de duas variáveis.
2 + 3 Nessa expressão, o valor retornado é a soma dos valores dados, isto é, 5.

Subtração Na matemática, representada pelo sinal – e, em expressões em termos computacionais, pelo mesmo sinal.

A – B Expressão que simboliza a subtração do valor de duas variáveis.
3 – 2 Nessa expressão, o valor retornado é o resto, isto é, 1.

Multiplicação Na matemática, representada pelos sinais × ou . e, em expressões em termos computacionais, pelo sinal *.

B * D Expressão que simboliza a multiplicação do valor de duas variáveis.
3 * 2 Nessa expressão, o valor retornado é o produto dos valores dados, isto é, 6.

Divisão Na matemática, representada pelo sinal ÷ e, em expressões computacionais, pelo sinal /.

A / B Expressão que simboliza a divisão do valor de duas variáveis.
6 / 2 Nessa expressão, o valor retornado é a divisão dos valores dados, que, no caso, será equivalente a 3.
5 / 2 Nessa expressão, o valor retornado é a divisão dos valores dados, que, no caso, será equivalente a 2.5.

OBSERVAÇÃO

- *Normalmente, as linguagens de programação assumem que a divisão é uma operação que retorna um valor REAL. Atenção especial, portanto, para variáveis que receberão o resultado de uma divisão.*

- *Em algumas linguagens, quando se divide dois números inteiros, o resultado será um inteiro.*

Exponenciação Na matemática, representada pela base e por um expoente e em expressões em termos computacionais pelo sinal (** ou ^) mais o número que se quer elevar.

A ** 2 Expressão que simboliza o valor da variável ao quadrado.
3 ^ 2 Nessa expressão, o valor retornado é o resultado da exponenciação do valor 3 ao quadrado (2) que, no caso, será equivalente a 9.
2 ** 3 Nessa expressão, o valor retornado é o resultado da exponenciação do valor 2 ao cubo (3), que no caso será equivalente a 8.00.

OBSERVAÇÃO

Normalmente, as linguagens oferecem um dos operadores citados, mas usaremos os dois e a diferença será explicada a seguir:

- ** - *exponenciação com resultado **Real***
- ^ - *exponenciação com resultado **Inteiro**, fazendo* arredondamento.

8 ** 3 A resposta seria 512.00
8 ^ 3 A resposta seria 512
8.5 ** 3 A resposta seria 614.125
8.5 ^ 3 A resposta seria 614

Radiciação pela potência

$\sqrt[índice]{radicando} = radicando^{1/índice}$

Exemplo: $\sqrt[3]{512} = 512^{1/3}$ -> 512**(1/3)

% – resto Em outras linguagens, conhecido como **mod.** É usado em expressões em termos computacionais quando se deseja encontrar o resto da divisão de *dois números inteiros*.

K % Y Expressão que simboliza a intenção de achar o resto da divisão do valor da variável K pelo valor da variável Y.
5 % 2 Nessa expressão, o valor retornado é o resto da divisão do primeiro pelo segundo número, que, no caso, será equivalente a 1.

7 % 4	Nessa expressão, o valor retornado é o resto da divisão do primeiro pelo segundo número, que, no caso, será equivalente a 3.

div – divisão inteira É usada em expressões em termos computacionais quando se deseja encontrar o quociente da divisão de dois números inteiros.

A **div** B	Expressão que simboliza a intenção de achar o valor do divisor na divisão do valor da variável A pelo valor da variável B.
5 **div** 2	Nessa expressão, o valor retornado é o coeficiente da divisão do primeiro pelo segundo número, que, no caso, será equivalente a 2.

↳ *O operador **div** necessita que, antes e depois dele, pressionemos a barra de espaço.*

Relacional

Uma expressão relacional, ou simplesmente relação, é uma comparação realizada entre dois valores de mesmo *tipo básico*. Estes valores são representados na relação através de constantes, variáveis ou expressões aritméticas.

Como exemplos de operadores relacionais matematicamente conhecidos temos:

Operador	Matemática	Usaremos
Igual	=	==
Diferente	≠	< >
Maior	>	>
Menor que	<	<
Maior ou Igual a	≥	>=
Menor ou Igual a	≤	<=

Como exemplos de expressões relacionais temos:

A < > B	A diferente de B
X == 1	X igual a 1
7 > 6	7 maior que 6
8 < 9	8 menor que 9
1 <= Y	1 menor ou igual ao valor da variável Y
4 >= W	4 maior ou igual ao valor da variável W

Lógica ou booleana

Denomina-se expressão lógica a expressão cujos operadores são lógicos e cujos operandos são relações, constantes e/ou variáveis do tipo lógico.

Como exemplo de operadores lógicos, matematicamente conhecidos temos:

Operador	Matemática	Usaremos
conjunção	e	&&
disjunção	ou	||
negação	nao	!

Tabela verdade do operador &&

Suponha duas perguntas feitas a quatro pessoas. Se a resposta do candidato for falsa, deverá falar 0, caso contrário, falará 1.

Suponha também que só será chamado para entrevista o candidato que dominar as duas linguagens.

Você conhece a linguagem C?	Você conhece a linguagem PASCAL?	Saída
0	0	0
0	1	0
1	0	0
1	1	1

Nesse exemplo, somente o quarto candidato seria chamado para a entrevista, pois o operador && (e) só considera a expressão como verdadeira se todas as expressões testadas forem verdadeiras.

Tabela verdade do operador ||

Suponha duas perguntas feitas a quatro pessoas. Se a resposta do candidato for falsa, deverá falar 0, caso contrário, falará 1.

Suponha também que será chamado para entrevista o candidato que dominar pelo menos uma linguagem.

Você conhece a linguagem C++?	Você conhece a linguagem JAVA?	Saída
0	0	0
0	1	1
1	0	1
1	1	1

Nesse exemplo, somente o primeiro candidato *não* seria chamado para a entrevista, pois o operador || (ou) considera a expressão como verdadeira se *pelo menos* uma expressão testada for verdadeira.

OBSERVAÇÃO

1. A seguir, relacionamos os critérios de precedência dos operadores. Lembre-se de que algumas linguagens não obedecem a estes critérios.

2. Se precisarmos alterar esta hierarquia, usaremos os parênteses.

Hierarquia	
primeiro	*parênteses e funções*
segundo	*potência e resto*
terceiro	*multiplicação e divisão*
quarto	*adição e subtração*
quinto	*operadores relacionais*
sexto	*operadores lógicos*

Tabela verdade do operador !

Observe a tabela a seguir e as afirmativas:

- A cor da camisa A não é azul.
- A cor da camisa B não é amarela.

Camisa	Cor	SAÍDA
A	Azul	falso
B	Verde	verdadeiro

O operador ! (não) inverte a saída.

Considere *a*, *b* e *c* variáveis numéricas, e cor uma variável string. Como exemplos de expressões lógicas temos:

a + b == 0 && c < > 1

Essa expressão verifica se o resultado da soma dos valores das variáveis **a** e **b** é igual a **0** e(&&) se o valor da variável **c** é diferente de **1**. O resultado será considerado **verdadeiro** se as *duas* expressões relacionais foram **verdadeiras**.

Cor == "azul" || a * b > c

Essa expressão verifica se o conteúdo armazenado na variável **cor** é azul **ou** (||) se o resultado do produto dos valores variáveis **a** e **b** é maior do que o valor armazenado na variável **c**. O resultado será considerado **verdadeiro** se, *pelo menos uma* das expressões relacionais for **verdadeira**.

O resultado obtido de uma avaliação de uma expressão lógica é sempre um valor lógico, isto é, **falso** ou **verdadeiro**. Por esse motivo, pode-se considerar uma única relação como sendo uma expressão lógica.

FUNÇÕES

O conceito de função em termos computacionais está intimamente ligado ao conceito de função (ou fórmula) matemática, onde um conjunto de variáveis e constantes numéricas relaciona-se por meio de operadores, compondo uma fórmula que, uma vez avaliada, resulta num valor. As expressões se dividem em:

Numéricas

Funções numéricas são aquelas cujo resultado da avaliação é do tipo numérico, seja ele inteiro ou real. Somente podem ser efetuadas entre números propriamente apresentados ou variáveis numéricas.

Como exemplos de funções numéricas temos:

pi Função que resulta no valor 3.14159265. Sem argumentos.

sen(x) Função que resulta no valor do seno de um ângulo qualquer em radianos.

↪ *Antes de aplicar a função **sen**(ang), deve-se transformar o Ângulo em Graus para* Ângulo Radiano *com a seguinte fórmula matemática: ang * 3.14159265 / 180 (ângulo multiplicado por 3.14159265 e o resultado dividido por 180). E logo depois se pode aplicar a função.*

A constante 3.14159265 será predefinida: **pi**.

Logo, teremos: angrad **<-** ang * **pi** / 180;
imprima sen(angrad);

Dessa forma, podemos observar que somente usamos a função ***sen(x)*** depois de transformar o ângulo, dado em graus, em ângulo radiano.

↪ *Normalmente, as linguagens de programação assumem que a função **sen**( ) é uma função que retorna um valor REAL. Atenção especial, portanto, para variáveis que receberão como conteúdo expressões que envolvam essa função.*

cos(x) Função que resulta no valor do co-seno de um de um ângulo qualquer em radianos.

Logo, teremos: angrad **<-** ang * **pi** / 180;
imprima cos(angrad);

↪ *Antes de aplicar a função **cos**(ang), deve-se transformar o Ângulo em Graus para* Ângulo Radiano *com a seguinte formula matemática: ang * 3.14159265/180 (ângulo multiplicado por 3.14159265 e o resultado dividido por 180). E logo depois se pode aplicar a função.*

Dessa forma, podemos observar que somente usamos a função ***cos(x)*** depois de transformar o ângulo, dado em graus, em ângulo radiano.

↪ *Normalmente, as linguagens de programação assumem que a função **cos**( ) é uma função que retorna um valor REAL. Atenção especial, portanto, para variáveis que receberão como conteúdo expressões que envolvam essa função.*

tan(x) Função que resulta no valor da tangente de um ângulo qualquer em radianos.

Logo, teremos: angrad **<-** ang * **pi** / 180;
imprima tan(angrad);

↳ *Antes de aplicar a função **tan**(ang), deve-se transformar o Ângulo em Graus para Ângulo Radiano com a seguinte fórmula matemática: ang * 3.14159265/180 (ângulo multiplicado por 3.14159265 e o resultado dividido por 180). E logo depois se pode aplicar a função.*

Dessa forma, podemos observar que somente usamos a função ***tan(x)*** depois de transformar o ângulo, dado em graus, em ângulo radiano.

↳ *Normalmente, as linguagens de programação assumem que a função **tan**( ) é uma função que retorna um valor REAL. Atenção especial, portanto, para variáveis que receberão como conteúdo expressões que envolvam essa função.*

abs(x) Função que resulta no valor absoluto de um número qualquer.

abs(7) Neste caso, a resposta fornecida seria 7
abs (–7) Neste caso, a resposta fornecida seria 7

exp(x) Função que resulta no valor do número e (base do logaritmo neperiano) elevado a um número qualquer.

exp (3) Nesse caso, seria o mesmo que e^3 -> 2.71828182846 ** 3
exp (2) Nesse caso, seria o mesmo que e^2 -> 2.71828182846 ** 2

Dessa forma, podemos observar que a função **exp(x)** se refere à base **e** (base do logaritmo neperiano: **2.71828182846**) elevada ao número citado entre os parênteses da função **exp(x).**

↳ *Normalmente, as linguagens de programação assumem que a função **exp**( ) é uma função que retorna um valor REAL. Atenção especial, portanto, para variáveis que receberão como conteúdo expressões que envolvam essa função.*

log(x) Função que resulta no valor do logaritmo neperiano de um número qualquer.

log(3) Nesse caso, seria: 1.09861

Dessa forma, podemos observar que o logaritmo ao qual a função **log(x)** se refere é sempre denominado pela base **e.**

↳ *Na prática, poderá ser necessário calcular o logaritmo em outra base e, para isso, você deverá fazer a conversão entre bases logarítmicas, usando a seguinte propriedade:*

$$\log_B^A = \frac{\log_x^A}{\log_x^B}$$

*Se considerarmos que a base dos logaritmos naturais (**e**) será usada, teremos:*

***log**(A) / **log**(B)*

↳ *Normalmente, as linguagens de programação assumem que a função **log**( ) é uma função que retorna um valor REAL. Atenção especial, portanto, para variáveis que receberão como conteúdo expressões que envolvam essa função.*

raiz(x) Função que resulta no valor da raiz quadrada de um número positivo.

raiz(4) Nesse caso, seria o mesmo que $\sqrt{4} = 2$
raiz(9) Nesse caso, seria o mesmo que $\sqrt{9} = 3$

Dessa forma, podemos observar que a função **raiz(x)** sempre fornece a raiz quadrada do argumento que sempre será positivo.

↳ *Normalmente, as linguagens de programação assumem que a função **raiz( )** é uma função que retorna um valor REAL. Atenção especial, portanto, para variáveis que receberão como conteúdo expressões que envolvam essa função.*

Funções de Conversão de Tipos

1) realint(número real**)** Função que converte um número real em inteiro.

realint(11.5) Nesse caso, retornaria 12.
realint(12.51) Nesse caso, retornaria 13.

↳ *Veja considerações sobre a função realint no Apêndice.*

2) **intreal**(número inteiro) Função que converte um número inteiro em real.

intreal(11) Nesse caso, retornaria 11.0 .
intreal(12) Nesse caso, retornaria 12.0 .

Caracter

Funções caracter são aquelas cujo resultado da avaliação é do tipo caracter. Somente podem ser efetuadas entre caracteres propriamente apresentados ou variáveis literais do tipo caracter.

Como exemplo de funções caracter temos:

strtam(string) Função que retorna número de caracteres de uma string.

strtam("rio") O resultado seria 3.
strtam(nome) Nesse caso o resultado será o tamanho do conteúdo da variável nome.

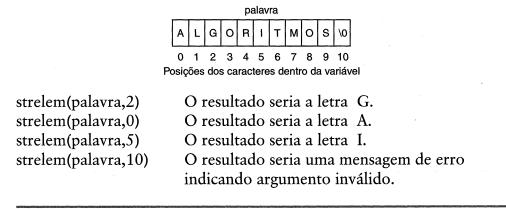
Dessa forma, podemos observar que a função **strtam(x)** sempre retorna o número de caracteres de uma string ou variável string.

↳ *Se o tamanho de uma variável string for armazenado em uma variável, o tipo dessa variável deverá ser **int**.*

---

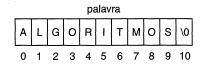
strelem(string, pos) Função que retorna o elemento da string que se encontra na posição indicada na função como *pos*.

Suponha a variável palavra

strelem(palavra,2) O resultado seria a letra G.
strelem(palavra,0) O resultado seria a letra A.
strelem(palavra,5) O resultado seria a letra I.
strelem(palavra,10) O resultado seria uma mensagem de erro indicando argumento inválido.

↳ *A variável ou constante pos, presente na função, representa a posição do caractere dentro da variável, porém não se esquecendo que a primeira posição é 0(zero).*

---

strprim(string) Função que retorna o primeiro elemento da string.

Suponha a variável palavra

strprim(palavra) O resultado seria a letra **A**, pois a função reconhece que o primeiro caractere se encontra na posição 0 (zero).

strnprim(string, n) Função que retorna os n primeiros elementos da string, incluindo a posição 0 (zero).

Suponha a variável palavra

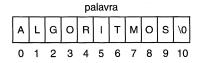
palavra

A	L	G	O	R	I	T	M	O	S	\0
0	1	2	3	4	5	6	7	8	9	10

strnprim(palavra, 2) O resultado seria **AL**, pois a função entende que os dois primeiros elementos estão nas posições 0 (zero) e 1 (um).

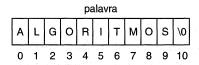
strnprim(palavra, 4) O resultado seria **ALGO**, pois a função entende que os quatro primeiros elementos estão nas posições 0 (zero), 1 (um), 2 (dois) e 3 (três).

strresto(string) Função que retorna todos os elementos da string, exceto o primeiro.

Suponha a variável palavra

strresto(palavra) O resultado seria **LGORITMOS**.

Nesse caso, a função reconhece que o primeiro elemento se encontra na posição 0 (zero).

strult(string) Função que retorna o último elemento da string.

Suponha a variável palavra

strult(palavra) O resultado seria a letra S.

strnresto(string, n) Função que retorna os elementos da string após os n primeiros.

Suponha a variável palavra

strnresto (palavra, 2) O resultado seria **GORITMOS.**

↳ *Nesse caso, a função reconhece que os dois primeiros elementos se encontram nas posições 0 (zero) e 1 (um).*

---

strcopia(string) Função que copia a string. Deverá ser usada com o comando de atribuição.

a <– "UNESA";	A string UNESA é armazenada na variável **a.**
b <– strcopia(a);	O conteúdo da variável **a** é copiado para variável **b.**
a <– "UNESA"; b <– a;	O resultado seria idêntico ao anterior.

OBSERVAÇÃO

Em algumas linguagens de programação, não é permitido usar o comando de atribuição a seguir:

b <– a ;

Mas, em compensação, há uma outra alternativa que possibilita uma variável string receber o conteúdo de outra variável string:

***strcopia**(string1,string2):*
strcopia(b, a);

strcomp(string1,string2) Função que resulta na comparação por ordem alfabética de duas strings (**string1** e **string2**) retornando:

"igual"	se forem iguais.
"menor"	se string1 vier antes de string2.
"maior"	se string1 vier depois de string2.

strcomp("maria", "maria")	Nesse caso, o valor retornado seria "igual" .
strcomp("aline", "alex")	Nesse caso, o valor retornado seria "maior" .
strcomp("carina" ,"simone")	Nesse caso, o valor retornado seria "menor" .
strcomp(a, b)	Nesse caso, seriam comparados os conteúdos das variáveis **a** e **b**. O resultado poderia ser: "maior", "igual" ou "menor".

↳ *Na maioria das linguagens, os resultados das comparações serão: **0** ou um número **negativo** ou um número **positivo**.*

Explicação:
Observe as figuras a seguir, cujas letras estão representadas pelos respectivos códigos ASCII:

a	l	i	n	e
97	108	105	110	101

a	l	e	x
97	108	101	119

quando se usa: ... strcomp("aline", "alex") ... , na verdade, compara-se o 1º código de aline com o 1º código de alex; como são iguais, compara-se o 2º código de aline com o 2º código de alex; como são iguais, compara-se o 3º código de aline com o 3º código de alex mas, nessa comparação (105 –101), o resultado foi maior do que zero (0), logo entende-se que aline, *na ordem alfabética*, vem *depois de* alex.

↳ *Normalmente, essa função será usada com estruturas de teste que serão vistas mais adiante.*

---

strconcat(string1, string2) Função que resulta na cópia do valor contido em uma string2 para o final da string1.

a <- “ANITA & ”; A string ANITA & é armazenada na variável **a.**
b <- “GUTO”; A string GUTO é armazenada na variável **b.**
c<- strconcat(a, b); A variável **c** recebe o conteúdo: ANITA&GUTO.

↳ *Os argumentos das funções strings deverão ser variáveis ou constantes. Funções não são permitidas.*

ATRIBUIÇÃO

É a principal forma de se armazenar um dado em uma variável. Esse comando permite que você forneça um valor a uma variável, onde o tipo desse valor tem de ser compatível com a variável.

identificador	<-	expressão	;

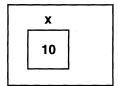
Legenda:

identificador	é o nome da variável à qual está sendo atribuído um valor.
<-	é o símbolo de atribuição, formado pelo sinais < e -.
expressão	pode ser uma expressão aritmética, uma expressão lógica ou literal cuja avaliação (resultado) é atribuída ao identificador (variável).
;	finaliza o comando.

Exemplo 1 : x **<-** 10 ;

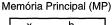
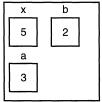
Como se lê ?
A variável x recebe o valor 10 ou x recebe o valor 10.

O que faz o computador ?
Nesse momento, na memória do computador, onde já estava alocado um espaço para a variável x (realizado na declaração de variáveis), essa variável recebe o valor 10.

Memória Principal (MP)

Exemplo 2 : x <- a + b ;

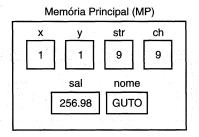
Como se lê?
A variável x recebe o resultado do conteúdo da variável a somado ao conteúdo da variável b ou x recebe o valor de a somado a b ou, ainda, x recebe a + b. *Lembre-se de que as operações aritméticas são realizadas pela UAL.*

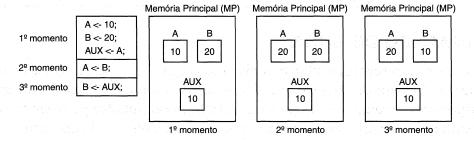
O que faz o computador?
Nesse momento, na memória do computador, onde já estava sendo alocado espaço para as variáveis a, b e x, o conteúdo da variável x vai receber a soma do conteúdo de a e b.

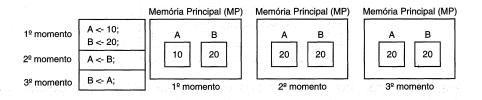
No comando de atribuição em que o valor é representado por uma expressão aritmética, lógica ou literal, estas devem ser avaliadas em primeiro lugar para que, então, o resultado obtido seja armazenado na variável.

Exemplo 3:

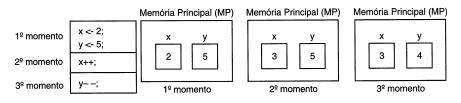
y <- 1;	y recebe o valor 1.
x <- y;	x recebe o conteúdo que está em y; mas como y vale 1, x vai receber 1, que é o conteúdo de y.
sal <- 256.98;	sal recebe o valor 256.98.
nome <- “GUTO”;	a variável nome recebe a string “GUTO”.
chr <- “g”;	a variável chr recebe o caractere “g”
str <- chr;	. str recebe o conteúdo de chr que é “g”.

Então, podemos resumir o exemplo acima como: x e y são duas variáveis inteiras; sal é uma variável do tipo real; nome é uma variável do tipo caractere; ch e str são variáveis do tipo char, e chave é uma variável do tipo lógica.

O mapa da memória nesse momento está assim. Podemos verificar que o tamanho do espaço na memória vai depender do tipo de variável.

Conclusão:
O comando de atribuição é muito importante em algoritmos. O próximo exemplo nos mostra isso.

Exemplo 4:

Qual o objetivo do algoritmo acima?
O conteúdo das variáveis A e B é trocado, isto é, no final do algoritmo, a variável **A** está com o valor 20 e a variável **B** está com o valor 10. Tivemos de utilizar uma variável auxiliar (AUX) que guarda o valor da variável A.

Para ficar mais clara a importância da variável auxiliar, observe a próxima figura e veja o que aconteceria se não tivéssemos feito uso dela:

Como pode ser visto, quando a variável **A** recebe o valor de **B**, a variável A perde o seu valor 10, recebendo o valor 20; depois, quando B receber o valor de A, B vai receber o valor 20 (novo valor de A). No final as variáveis **A** e **B** vão ter o mesmo valor 20.

Exemplo 5:

↳ *Os operadores ++ e -- são operadores de* incremento *e* decremento. *Eles são usados para realizar operações de adição e subtração e são similares à atribuição para variáveis* ***inteiras****.*
x++ ; É equivalente a: x <- x +1 ;
y-- ; É equivalente a: y <- y -1 ;

COMANDO DE SAÍDA

É o comando responsável por enviar um resultado, uma informação ao usuário. O valor de cada variável é buscado na memória e inserido em um dispositivo de saída. Através desse comando o computador pode emitir os resultados e outras mensagens para o usuário através da tela do computador ou uma impressora.

imprima expressão ou variável ou constantes ;

algoritmo 11

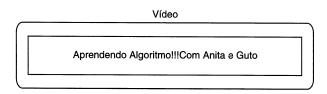
```
prog imp1
 imprima "Aprendendo Algoritmo!!!";
fimprog
```

Vídeo

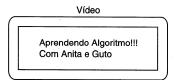
Aprendendo Algoritmo!!!

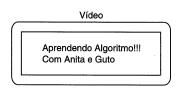
Esse algoritmo faz com que seja exibida, na tela do computador, a mensagem: Aprendendo Algoritmo!!!

algoritmo 12

```
prog imp2
 imprima "Aprendendo Algoritmo !!!";
 imprima "Com Anita e Guto";
fimprog
```

Embora tenhamos duas linhas de comandos, o que nos levaria a pensar que teríamos duas linhas no vídeo, o interpretador só alimenta linha se assim especificarmos através do símbolo **\n**. Esse símbolo é uma string e deverá vir entre aspas.

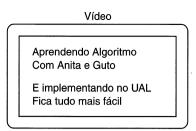
algoritmo 13

```
prog imp3
  imprima "Aprendendo Algoritmo!!!";
  imprima "\nCom Anita e Guto";
fimprog
```

algoritmo 14

```
prog imp4
   imprima "Aprendendo Algoritmo !!!\n ";
   imprima "Com Anita e Guto";
fimprog
```

Observe que o símbolo **\n** poderá ser colocado ao final da linha anterior ou no início da próxima linha que produzirá o mesmo efeito mas, lembre-se, *sempre entre aspas*.

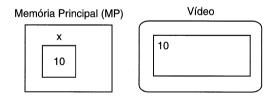
algoritmo 15

```
prog imp5
 imprima "Aprendendo Algoritmo \n Com Anita e Guto\n\n E implementando no UAL\n
        Fica muito mais fácil!! "; # digite tudo na mesma linha
fimprog
```

***Quando aparecer comentários do tipo: #** continuacao (ou **#** continuacao da linha anterior, significa que você deverá digitar tudo em uma só linha. **Não pressione enter**).*

Observe que podemos, usando um único comando **imprima** e fazendo uso do símbolo **\n**, mostrar várias mensagens em várias linhas, inclusive deixando linha em branco quando colocamos **\n\n**.

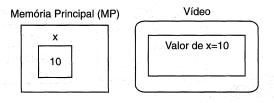
algoritmo 16

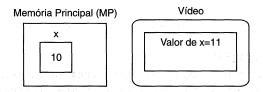
```
prog imp6
 int x;
 x <- 10;
 imprima  x ;
fimprog
```

Esse trecho é de muita importância pois **x** recebe o valor 10. Como já vimos, na memória do computador, existe uma variável chamada **x** e o valor 10 seria armazenado dentro da variável. Quando o comando **imprima** é executado, o valor de x, que está na memória do computador, é exibido pelo comando **imprima** no vídeo.

algoritmo 17

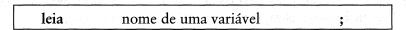
```
prog imp7
 int x;
 x <- 10;
 imprima  "Valor de x = ", x;
fimprog
```

Esse trecho permite a exibição de uma mensagem e do conteúdo de uma variável na tela do computador.

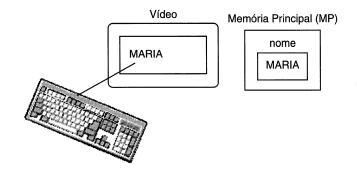
algoritmo 18

```
prog imp8
 int x;
 x <- 10;
 imprima  "Valor de x = ", x+1;
fimprog
```

Esse trecho é bem parecido com o anterior. O conteúdo da variável **x** é copiado da memória e acrescido de um, sendo impresso, após a string, sem alterar o valor de x na MP.

COMANDO DE ENTRADA

É o comando que permite que o usuário digite dados, possibilitando um "diálogo com o computador". O dado digitado é armazenado temporariamente em um registrador e, depois, copiado para a posição de memória indicada no comando. Lembre-se de que o nome de uma variável representa uma posição de memória.

Sintaxe:

Exemplo 1: **leia** nome;

Como se lê?
Leia um valor para a variável nome.

O que faz o computador?
O computador fica "esperando" o usuário digitar um dado; neste exemplo, um nome: Maria. A variável nome, tendo sido declarada como string, recebe o valor Maria. Para facilitar, o dado digitado é sempre mostrado na tela.

↳ *Veja considerações sobre o comando de entrada para o interpretador sugerido no Apêndice I.*

Observe o algoritmo a seguir que envolve a maioria das funções numéricas:

algoritmo 19

```
prog teste
imprima "raiz: ",raiz(64);
imprima "\nraiz com exp e log e realint: ",realint(exp(1/2*log(64)));
imprima "\nraiz com exp e log sem realint: ",exp(1/2*log(64));
imprima "\n", formatar(sen(45*pi/180)+0.0001,3);
imprima "\npotencia com exp e log e formatar: ",formatar(exp(
        3*log(8))+0.001,3);#continuacao
imprima "\npotencia com exp e log sem formatar: ",exp(3*log(8));
imprima "\npotencia com operador ** e formatar: ",formatar(8**3,3);
imprima "\nraiz cubica: ",exp(1/3*log(8));
imprima "\nabsoluto: ",abs(-8);
imprima "\nabsoluto: ",abs(8);
imprima "\nconvertendo para inteiro 5.5: ",realint(5.5);
imprima "\nconvertendo para inteiro 6.5: ",realint(6.5);
imprima "\nconvertendo para inteiro 6.5 + 0.0001: ",realint(6.5+0.0001);
imprima "\nconvertendo para inteiro 5.45: ",realint(5.45);
imprima "\nconvertendo para inteiro 5.51: ",realint(5.51);
imprima "\nconvertendo para real 87: ",intreal(87);
imprima "\nconvertendo para int 3/4: ",realint(3/4),"\n";
fimprog
```

VÍDEO

```
raiz: 8.0
raiz com exp e log e realint: 8
raiz com exp e log sem realint: 7.999999999999998
0.707
potencia com exp e log e formatar: 512.000
potencia com exp e log sem formatar: 511.9999999999995
potencia com operador ** e formatar: 512.000
raiz cubica: 1.9999999999999998
absoluto: 8
absoluto: 8
convertendo para inteiro 5.5: 6
convertendo para inteiro 6.5: 6
convertendo para inteiro 6.5 + 0.0001: 7
convertendo para inteiro 5.45: 5
convertendo para inteiro 5.51: 6
convertendo para real 87: 87.0
convertendo para int 3/4: 1
```

Observe o algoritmo a seguir que envolve a maioria das funções strings:

algoritmo 20

```
prog funcoes
 string c,c1,d,d1;
 mprima "\ndigite palavra 1: ";
 leia c;
 imprima "\ndigite palavra 2: ";
 leia c1;
 imprima "\n",strtam(c) ;
 imprima "\nconcatenando: ",strconcat(c,c1) ;
 d <- strcopia(c) ;
 imprima "\nE O CONTEUDO DE D: ",d;
 d1<-strconcat(c,c1) ;
 imprima "\nconcatenacao: ",d1;
 imprima "\nprimeiro caractere: ",strprim(c);
 imprima "\nultimo caractere: ",strult(c);
 imprima "\ntodos menos o primeiro: ",strresto(c);
 imprima "\no terceiro elemento: ",strelem(c,3); # sairá o 4º caractere
 imprima "\nos tres primeiros elementos: ",strnprim(c,3);
 imprima "\nos tres ultimos elementos: ",strnresto(c,3);
 imprima "\n";
fimprog
```

VÍDEO
digite palavra:TESTANDO digite palavra2:UAL tamanho da 1 palavra: 8 concatenando sem armazenar:TESTANDOUAL o conteudo de d(variavel que teve valor copiado de c: TESTANDO o conteudo de e(variavel que teve valor atribuido de c: TESTANDO concatenacao com armazenamento:TESTANDOUAL primeiro caractere:T ultimo caractere:L todos menos o primeiro:ESTANDOUAL o terceiro elemento:S os tres primeiros elementos:TES os tres ultimos elementos:UAL

Testando Hierarquia

algoritmo 21

```
prog teste
 imprima "\nTESTANDO HIERARQUIA\n";
 imprima "\n12 + 5 /2 é igual a: ", 12 + 5 / 2;
 imprima "\nÉ DIFERENTE DE (12 + 5)/2 que é igual a: ", (12 + 5)/2," ,
        logo / tem HIERARQUIA MAIOR do que + ou -";#continuacao
 imprima "\n\n64**1/4 é igual a: ",64**1/4 ;
 imprima "\nÉ DIFERENTE de 64**(1/4)que é igual a: ",64**(1/4)," logo **
        tem HIERARQUIA MAIOR do que * ou / " ;#continuacao
 imprima "\n\n3*7 % 5 é igual a: ", 3*7 % 5;
 imprima "\nÉ DIFERENTE de (3 * 7) % 5 que é igual a: ",(3 * 7) % 5, "
        logo % tem HIERARQUIA MAIOR do que * " ;#continuacao
 imprima "\n\n3 * 7 div 5 é igual a: ", 3*7 div 5;
 imprima "\nÉ IGUAL a (3 * 7) div 5 : ",(3 * 7) div 5," logo div tem a
        MESMA HIERARQUIA da * ou / ";#continuacao
 imprima "\n";
fimprog
```

VÍDEO
TESTANDO HIERARQUIA 12 + 5 /2 é igual a: 14.5 É DIFERENTE DE (12 + 5) /2, que é igual a: 8.5 s; logo, / tem HIERARQUIA MAIOR do que + ou - 64**1/4 é igual a: 16.0 É DIFERENTE de 64**(1/4); que é igual a: 2.8284271247461903; logo ** tem HIERARQUIA MAIOR do que * ou / 3*7 % 5 é igual a: 6 É DIFERENTE de (3 * 7) % 5, que é igual a: 1; logo, % tem HIERARQUIA MAIOR do que * 3 * 7 div 5 é igual a: 4 É IGUAL a (3 * 7) div 5 : 4; logo, div tem a MESMA HIERARQUIA de * ou /

Uma operação de divisão não pode ser um dos operandos de uma expressão com %, pois o operador % tem hierarquia maior do que a divisão. Logo, 6%2 é executado primeiro e, quando for feita a divisão, o divisor será zero (0). Coloque a operação de divisão entre parênteses para ser executada primeiro.

Um número real pode ser usado como operando de div, porém será feito arredondamento. Isso é *desaconselhável*, pois acaba com a filosofia de div.

algoritmo 22

```
prog teste
 imprima "\nTESTANDO HIERARQUIA";
 imprima "\n\n18/6 % 2 é igual a: ", 18 / 6 % 2;
 imprima "\nUma operação de divisão fora de parênteses não pode ser um
        dos operandos de uma expressao com  %.";#continuacao
 imprima "\n\n20 / 4 div 2 é igual a: ", 20 / 4 div 2;
 imprima "\nÉ IGUAL a (20 / 4) div 2 : ",(20 / 4) div 2," logo div tem a
        MESMA HIERARQUIA ";#continuacao
 imprima " da  /. ";
 imprima "\n\n30 / 4 div 2 é igual a: ", 30/4 div 2;
 imprima "\nÉ IGUAL a (30 / 4) div 2 : ",(30 / 4) div 2," logo div tem
        a MESMA HIERARQUIA ";#continuacao
 imprima " da / ";
```

```
 imprima "\n\n7. div 4: ",7. div 4;
 imprima "\n7  div 4: ",7 div 4;
 imprima "\n6. div 4: ",6. div 4;
 imprima "\n6  div 4: ",6 div 4;
 imprima "\n";
fimprog
```

VÍDEO

```
TESTANDO HIERARQUIA

18/6 % 2 é igual a: Infinity
Uma operação de divisão fora de parênteses não pode ser um dos
operandos de uma expressão com %.

20 / 4 div 2 é igual a: 2
É IGUAL a (20 / 4) div 2 : 2; logo, div tem a MESMA HIERARQUIA da /.

30 / 4 div 2 é igual a: 4
É IGUAL a (30 / 4) div 2 : 4; logo, div tem a MESMA HIERARQUIA da /

7. div 4: 2
7  div 4: 1
6. div 4: 2
6  div 4: 1
```

Muita atenção para o uso dos operadores % e div numa expressão.
Se % vier antes, não tem problema.

algoritmo 23

Entrar com um número inteiro de 3 casas e imprimir o algarismo da casa das dezenas.

```
prog teste
 int a,d;
 imprima "\nDigite numero de tres casas: ";
 leia a;
 d<-a % 100 div 10;
 imprima "\nalgarismo da casa das dezenas: ",d;
 imprima "\n";
fimprog
```

VÍDEO
Digite numero de tres casas:135 algarismo da casa das dezenas: 3

Se a idéia é efetuar a operação com div antes, teremos problema, pois o operador % tem hierarquia maior que div; logo, primeiro será operado 10%10, cujo o resultado é zero (0). Isso impossibilita a operação com div, uma vez que não se pode dividir por 0.

algoritmo 24

```
prog teste
 int a,d;
 imprima "\nDigite numero de tres casas: ";
 leia a;
 d<-a div 10 % 10;
 imprima "\nalgarismo da casa das dezenas: ", d;
 imprima "\n";
 fimprog
```

VÍDEO
Digite numero de tres casas:135 Floating point exception (core dumped)

Lembre-se sempre disto: quando você montar uma expressão com div e %, se div vier antes de %, coloque parênteses para priorizar uma operação de hierarquia menor.

algoritmo 24

Sem problema

```
prog teste
 int a,d;
 imprima "\nDigite numero de tres casas: ";
 leia a;
 d<-(a div 10) % 10;
 imprima "\nalgarismo da casa das dezenas: ",d;
 imprima "\n";
fimprog
```

VÍDEO
Digite numero de tres casas:135 algarismo da casa das dezenas: 3

Algumas aplicações com % e div

algoritmo 25

Entrar com uma data no formato ddmmaa e imprimir: dia, mês e ano separados.

```
prog teste
 int data,dia,mes,ano;
 imprima "\nDigite data no formato ddmmaa: ";
 leia data;
 dia<-data div 10000;
 mes<-data % 10000 div 100;
 ano<-data %100;
 imprima "\nDIA: ",dia;
 imprima "\nMES: ",mes;
 imprima "\nANO: ",ano;
 imprima "\n";
fimprog
```

VÍDEO
Digite data no formato DDMMAA:230389 DIA:23 MES:3 ANO:89

algoritmo 26

Entrar com uma data no formato ddmmaa e imprimir no formato mmddaa.

```
prog teste
 int data,dia,mes,ano,ndata;
 imprima "\nDigite data no formato DDMMAA: ";
 leia data;
 dia<-data div 10000;
 mes<-data % 10000 div 100;
```

```
 ano<-data %100;
 ndata<-mes *10000 +dia*100+ano;
 imprima "\nDIA: ",dia;
 imprima "\nMES: ",mes;
 imprima "\nANO: ",ano;
 imprima "\n\nDATA NO FORMATO  MMDDAA: ",ndata;
 imprima "\n";
fimprog
```

VÍDEO
Digite data no formato DDMMAA:251201 DIA:25 MES:12 ANO:1 DATA NO FORMATO MMDDAA:122501

Os operadores ++ e --

algoritmo 27

```
prog xx
 int x,y;
 x<-2;
 y<-5;
 imprima "\nx = ",x;
 x++;
 imprima "\n\ny = ",y;
 y--;
 imprima "\n\nnovo valor de x = ",x;
 imprima "\n\nnovo valor de y = ",y;
 imprima "\n\n";
fimprog
```

VÍDEO
x = 2 y = 5 novo valor de x = 3 novo valor de y = 4

Só reforçando

- Todas as palavras reservadas são escritas com letras minúsculas.
- O operador de atribuição deverá ser formado pelo sinal **<** seguido do sinal –, ficando: **<-** .
- Os identificadores (nome do algoritmo e das variáveis deverão começar por uma **letra**, e os demais caracteres por **letra** ou **algarismo**).
- Os comandos: **imprima, leia, atribuição** e as **declarações de variáveis** terminam com ; .
- Os comandos **prog** e **fimprog** não têm ; .

EXERCÍCIOS – LISTA 1
LEIA, IMPRIMA, ATRIBUIÇÃO E FUNÇÕES

algoritmo 28

Imprimir a mensagem: "É PRECISO FAZER TODOS OS ALGORITMOS PARA APRENDER".

```
prog lea1
 imprima "\nE PRECISO FAZER TODOS OS ALGORITMOS PARA APRENDER ";
 imprima "\n";
fimprog
```

↳ ***imprima "\n"****; será sempre colocada para que o prompt não fique na mesma linha da última impressão, uma vez que o cursor não desce automaticamente.*

algoritmo 29

Imprimir seu nome.

```
prog lea2
 imprima "\n <seu nome>";
 imprima "\n";
fimprog
```

algoritmo 30

Criar um algoritmo que imprima o produto entre 28 e 43.

```
prog lea3
 int  produto;
 produto <- 28 * 43;
 imprima "\nO produto entre os dois e: ", produto;
 imprima "\n";
fimprog
```

algoritmo 31

Criar um algoritmo que imprima a média aritmética entre os números 8, 9 e 7.

```
prog lea4
 real ma;
 ma <- ( 8 +  9 + 7 ) / 3;
 imprima "\nA media aritmetica e: ", ma;
 imprima "\n";
fimprog
```

algoritmo 32

Ler um número inteiro e imprimi-lo.

```
prog lea5
 int num;
 imprima "\n entre com um numero: ";
 leia num;
 imprima "\nnumero : ", num;
 imprima "\n";
fimprog
```

algoritmo 33

Ler dois números inteiros e imprimi-los.

```
prog lea6
 int num1, num2;
 imprima "\n entre com um numero: ";
 leia num1;
 imprima "\n entre com outro numero: ";
 leia num2;
 imprima "\nnumero 1 : ", num1;
 imprima "\nnumero 2 : ", num2;
 imprima "\n";
fimprog
```

algoritmo 34

Ler um número inteiro e imprimir seu sucessor e seu antecessor.

```
prog lea7
 int numero, suc, ant;
 imprima "\n entre com um numero: ";
 leia numero;
 ant <- numero -1;
 suc <- numero +1;
 imprima "\no sucessor e ƀ", suc,"ƀ o antecessor e ƀ", ant;
 imprima "\n";
fimprog
```

algoritmo 35

Ler nome, endereço e telefone e imprimi-los.

```
prog lea8
 string nome, endereco, telefone;
 imprima "\nentre com nome: ";
 leia nome;
 imprima "\nentre com endereco: ";
 leia endereco;
 imprima "\nentre com  telefone: ";
 leia telefone;
 imprima "\n\n\n";
 imprima "\nNome    : ",nome;
 imprima "\nEndereco: ",endereco;
 imprima "\nTelefone: ",telefone;
 imprima "\n";
fimprog
```

algoritmo 36

Ler dois números inteiros e imprimir a soma. Antes do resultado, deverá aparecer a mensagem: Soma.

```
prog lea9
 int num1, num2, soma;
 imprima "\n entre com um numero: ";
 leia num1;
 imprima "\n entre com outro numero: ";
 leia num2;
 soma <- num1 + num2;
 imprima "\nSoma: ", soma;
 imprima "\n";
fimprog
```

algoritmo 37

Ler dois números inteiros e imprimir o produto.

```
prog lea10
 int num1, num2, prod;
 imprima "\n entre com um numero: ";
 leia num1;
 imprima "\n entre com outro numero: ";
 leia num2;
 prod <- num1 * num2;
 imprima "\nproduto: ", prod;
 imprima "\n";
fimprog
```

algoritmo 38

Ler um número real e imprimir a terça parte deste número.

```
prog leal1
 real num;
 imprima "\nentre com um numero com ponto: ";
 leia num;
 imprima "\na terça parte e: ", num/3;
 imprima "\n";
fimprog
```

algoritmo 39

Entrar com dois números reais e imprimir a média aritmética com a mensagem "média" antes do resultado.

```
prog leal2
 real nota1, nota2, media;
 imprima "\ndigite 1a nota: ";
 leia nota1;
 imprima "\ndigite 2a  nota: ";
 leia nota2;
 media <- ( nota1 + nota2)/2;
 imprima "\nmedia: ", media;
 imprima "\n";
fimprog
```

algoritmo 40

Entrar com dois números inteiros e imprimir a seguinte saída:

```
dividendo:
divisor:
quociente:
resto:
prog leal3
 int quoc, rest, val1, val2;
 imprima "\nentre com o dividendo: ";
 leia val1;
 imprima "\nentre com divisor: ";
 leia val2;
 quoc <- val1 div val2;
 rest <- val1 % val2;
 imprima "\n\n\n";
 imprima "\ndividendo  : ", val1;
 imprima "\ndivisor    : ", val2;
 imprima "\nquociente  : ", quoc;
 imprima "\nresto      : ", rest;
 imprima "\n";
fimprog
```

algoritmo 41

Entrar com quatro números e imprimir a média ponderada, sabendo-se que os pesos são respectivamente: 1, 2, 3 e 4.

```
prog lea14
 real a, b, c,d, mp;
 imprima "\nentre com 1 numero: ";
 leia a;
 imprima "\nentre com 2 numero: ";
 leia b;
 imprima "\nentre com 3 numero: ";
 leia c;
 imprima "\nentre com 4 numero: ";
 leia d;
 mp <- (a*1 + b*2 + c*3 + d*4)/10;
 imprima "\nmedia ponderada: ", mp;
 imprima "\n";
fimprog
```

algoritmo 42

Entrar com um ângulo em graus e imprimir: seno, co-seno, tangente, secante, co-secante e co-tangente deste ângulo.

```
prog lea15
 real angulo, rang;
 imprima "\ndigite um angulo em graus: ";
 leia angulo;
 rang <- angulo*pi /180;
 imprima "\nseno: ", sen(rang);
 imprima "\nco-seno: ", cos(rang);
 imprima "\ntangente: ", tan(rang);
 imprima "\nco-secante: ",1/ sen(rang);
 imprima "\nsecante: ", 1/cos(rang);
 imprima "\ncotangente: ", 1/ tan(rang);
 imprima "\n";
fimprog
```

Alguns ângulos que você digitar poderão não ter resposta em algumas funções, mas este problema será resolvido quando você aprender a estrutura de teste.

algoritmo 43

Entrar com um número e imprimir o logaritmo desse número na base 10.

```
prog lea16
 real num, logaritmo;
```

```
 imprima "\nentre com o logaritmando: ";
 leia num;
 logaritmo <- log(num) / log(10);
 imprima "\nlogaritmo: ", logaritmo;
 imprima "\n";
fimprog
```

algoritmo 44

Entrar com o número e a base em que se deseja calcular o logaritmo desse número e imprimi-lo.

```
prog lea17
 real num, base, logaritmo;
 imprima "\nentre com o logaritmando: ";
 leia num;
 imprima "\nentre com a base: ";
 leia base;
 logaritmo <- log(num) / log(base);
 imprima "\no logaritmo deƀ", num, "ƀna baseƀ",base, "ƀe:ƀ", logaritmo;
 imprima "\n";
fimprog
```

algoritmo 45

Entrar com um número e imprimir a seguinte saída:

```
numero:
quadrado:
raiz quadrada:
prog lea18
 real num, quad, raizquad;
 imprima "\ndigite numero: ";
 leia num;
 quad <-num ** 2;
 raizquad <- raiz(num);
 imprima "\nnumero: ", num;
 imprima "\nquadrado: ", quad;
 imprima "\nraiz quadrada: ", raizquad;
 imprima "\n";
fimprog
```

algoritmo 46

Fazer um algoritmo que possa entrar com o saldo de uma aplicação e imprima o novo saldo, considerando o reajuste de 1%.

```
prog lea19
 real saldo, nsaldo;
 imprima "\ndigite saldo: ";
```

```
 leia saldo;
 nsaldo <-saldo  * 1.01;
 imprima "\nnovo saldo: ",nsaldo;
 imprima "\n";
fimprog
```

algoritmo 47

Entrar com um número no formato CDU e imprimir invertido: UDC. (Exemplo: 123, sairá 321.) O número deverá ser armazenado em outra variável antes de ser impresso.

```
prog lea20
 int num, c, d, u, num1;
 imprima "\nentre com um número de 3 dígitos: ";
 leia num;
 c <- num div 100;
 d  <- num % 100 div 10;
 u  <- num % 10;
 num1 <- u*100 + d*10 + c;
 imprima "\nnúmero: ", num;
 imprima "\ninvertido: ", num1;
 imprima "\n";
fimprog
```

algoritmo 48

Antes de o racionamento de energia ser decretado, quase ninguém falava em quilowatts; mas, agora, todos incorporaram essa palavra em seu vocabulário. Sabendo-se que 100 quilowatts de energia custa um sétimo do salário mínimo, fazer um algoritmo que receba o valor do salário mínimo e a quantidade de quilowatts gasta por uma residência e calcule. Imprima:

- *o valor em reais de cada quilowatt*
- *o valor em reais a ser pago*
- *o novo valor a ser pago por essa residência com um desconto de 10%.*

```
prog lea21
 real sm, qtdade, preco, vp, vd;
 imprima "\nentre com o salário mínimo: ";
 leia sm;
 imprima "\nentre com a quantidade em quilowatt: ";
 leia qtdade;
 # divide por 7 para achar o preço de 100 Kw e por 100 para achar de 1 Kw
 preco <- sm /700;
 vp <- preco * qtdade;
 vd  <- vp * 0.9;
 imprima "\npreço do quilowatt: ", preco, "\n valor a ser pago: ", vp,
```

```
 "\n valor com desconto: ", vd;
 imprima "\n";
fimprog
```

algoritmo 49

Entrar com um nome e imprimir:

```
 todo nome:
 primeiro caractere:
 ultimo caractere:
 do primeiro ate o terceiro:
 quarto caractere:
 todos menos o primeiro:
 os dois ultimos:
prog lea22
 string nome;
 int n;
 imprima "\nentre com nome: ";
 leia nome;
 imprima "\ntodo nome: " , nome;
 imprima "\nprimeiro caractere: ", strprim(nome);
 imprima "\nultimo caractere: ", strult(nome);
 imprima "\nprimeiro ao terceiro caractere: ", strnprim(nome,3);
 imprima "\nquarto caractere: ", strelem(nome,3);
 imprima "\ntodos menos o primeiro: ", strresto ( nome);
 n <-strtam(nome) -2;
 imprima "\nos dois ultimos: " , strnresto(nome,n);
 imprima "\n";
fimprog
```

algoritmo 50

Entrar com a base e a altura de um retângulo e imprimir a seguinte saída:

```
 perimetro:
 area:
 diagonal:
prog lea23
 real  perimetro, area, diagonal, base, altura;
 imprima "\ndigite base: ";
 leia base;
 imprima "\ndigite altura: ";
 leia altura;
 perimetro <- 2*(base + altura);
 area <-base * altura;
 diagonal <- raiz(base**2 + altura**2);
 imprima "\nperimetro = " ,perimetro;
 imprima "\narea = ", area ;
```

```
 imprima "\ndiagonal = ", diagonal ;
 imprima "\n";
fimprog
```

algoritmo 51

Entrar com o raio de um círculo e imprimir a seguinte saída:

```
 perimetro:
 area:
prog lea24
 real  raio, perimetro, area;
 imprima "\ndigite raio: ";
 leia raio;
 perimetro <- 2* pi * raio;
 area <-pi *raio ** 2;
 imprima "\nperimetro : " , perimetro;
 imprima "\narea :  ", area ;
 imprima "\n";
fimprog
```

algoritmo 52

Entrar com o lado de um quadrado e imprimir:

```
 perimetro:
 area:
 diagonal:
prog lea25
 real lado, perimetro, area, diagonal;
 imprima "\ndigite o lado do quadrado: ";
 leia lado ;
 perimetro <- 4 * lado;
 area<- lado ** 2;
 diagonal <- lado * raiz(2);
 imprima "\nperimetro: ", perimetro;
 imprima "\narea: ", area;
 imprima "\ndiagonal: ", diagonal;
 imprima "\n";
fimprog
```

algoritmo 53

Entrar com os lados a, b, c de um paralelepípedo. Calcular e imprimir a diagonal.

```
prog lea26
 real a, b, c, diagonal;
 imprima "\nentre com a base: ";
 leia a;
 imprima "\nentre com a altura: ";
 leia b;
```

```
 imprima "\nentre com a profundidade: ";
 leia c;
 diagonal <-raiz( a**2 + b**2 + c**2 );
 imprima "\ndiagonal : ", diagonal;
 imprima "\n";
fimprog
```

algoritmo 54

Criar um algoritmo que calcule e imprima a área de um triângulo.

```
prog lea27
 real   a, b;
 imprima "\nEntre com a base: ";
 leia a;
 imprima "\nEntre a altura do um triângulo: ";
 leia b;
 imprima "\nArea = ", (a * b)/2;
 imprima "\n";
fimprog
```

algoritmo 55

Criar um algoritmo que calcule e imprima a área de um losango.

```
prog lea28
 real diagmaior, diagmenor, area;
 imprima "\nmedida da diagonal maior: ";
 leia  diagmaior;
 imprima "\nmedida da diagonal menor: ";
 leia  diagmenor;
 area <- (diagmaior * diagmenor)/2;
 imprima "\narea =", area;
 imprima "\n";
fimprog
```

algoritmo 56

Entrar com nome e idade. Imprimir a seguinte saída:

```
 nome:
 idade:
prog lea29
 string nome;
 int idade;
 imprima "\ndigite nome: ";
 leia nome;
 imprima "\ndigite idade: ";
 leia idade;
 # a linha abaixo é para dar uma separação entre a entrada e a saída
```

```
 imprima "\n\n";
 imprima "\nnome = ", nome;
 imprima "\nidade = ", idade;
 imprima "\n";
fimprog
```

algoritmo 57

Entrar com as notas da PR1 e PR2 e imprimir a média final:

- *truncada:*
- *arredondada:*

```
prog lea30
 real  pr1, pr2, mf;
 imprima "\ndigite pr1: ";
 leia pr1;
 imprima "\ndigite pr2: ";
 leia pr2;
 mf <- ( pr1 + pr2 ) / 2;
 imprima "\nmedia truncada = ", realint((mf- 0.5)+0.001);
 imprima "\nmedia arredondada = ", realint( mf+0.001);
 imprima "\n";
fimprog
```

VÍDEO	
digite pr1:2.3 digite pr2:3.7 media truncada = 3 media arredondada = 3 digite pr1:2.8 digite pr2:2.7 media truncada = 2 media arredondada = 3	digite pr1:7.9 digite pr2:8.1 media truncada = 8 media arredondada = 8 digite pr1:6.9 digite pr2:8.1 media truncada = 7 media arredondada = 8

algoritmo 58

Entrar com valores para xnum1, xnum2 e xnum3 e imprimir o valor de x, sabendo-se que:

$$X = xnum1 + \frac{xnum2}{xnum3 + xnum1} + 2(xnum1 - xnum2) + \log_2^{64}$$

```
prog lea31
 real xnum1, xnum2, xnum3, x;
 imprima "\nEntrar com 1 valor: ";
 leia xnum1;
 imprima "\nEntrar com 2 valor: ";
 leia  xnum2;
 imprima "\nEntrar com 3 valor: ";
 leia  xnum3;
 x <- xnum1 + xnum2 / (xnum3 + xnum1) + 2 *(xnum1 - xnum2) + log(64.)/
 log(2.);
 imprima "\nX = ", x;
 imprima "\n";
fimprog
```

algoritmo 59

Entrar com os valores dos catetos de um triângulo retângulo e imprimir a hipotenusa.

```
prog lea32
 real a,b,c;
 imprima "\nEntrar com 1 cateto: ";
 leia b;
 imprima "\nEntrar com 2  cateto: ";
 leia c;
 a <- raiz (b**2 + c**2);
 imprima "\nA hipotenusa e: ", a;
 imprima "\n";
fimprog
```

algoritmo 60

Entrar com a razão de uma PA e o valor do 1º termo. Calcular e imprimir o 10º termo da série.

```
prog lea33
 int dec, razao, termo;
 imprima "\nEntrar com o 1o termo: ";
 leia termo;
 imprima "\nEntrar com a razao: ";
 leia razao;
 dec <- termo + 9* razao;
 imprima "\nO 10 termo desta P.A. e: ", dec;
 imprima "\n";
fimprog
```

algoritmo 61

Entrar com a razão de uma PG e o valor do 1º termo. Calcular e imprimir o 5º termo da série.

```
prog lea34
 int quinto, razao, termo;
 imprima "\nEntre com o 1o termo: ";
 leia termo;
 imprima "\nEntre com a razao: ";
 leia razao;
 quinto <- termo * razao ^4;
 imprima "\nO 5o  termo desta P.G. e: ", quinto;
 imprima "\n";
fimprog
```

algoritmo 62

Em épocas de pouco dinheiro, os comerciantes estão procurando aumentar suas vendas oferecendo desconto. Faça um algoritmo que possa entrar com o valor de um produto e imprima o novo valor tendo em vista que o desconto foi de 9%.

```
prog lea35
 real preco, npreco;
 imprima "\ndigite valor do produto: ";
 leia preco;
 npreco <-preco * 0.91;
 imprima "\npreco com desconto: ",npreco;
 imprima "\n";
fimprog
```

algoritmo 63

Criar um algoritmo que efetue o cálculo do salário líquido de um professor. Os dados fornecidos serão: valor da hora aula, número de aulas dadas no mês e percentual de desconto do INSS.

```
prog lea36
 int na;
 real   vha, pd, td, sb, sl;
 imprima "\nhoras trabalhadas: ";
 leia  na ;
 imprima "\nvalor da hora-aula: ";
 leia  vha ;
 imprima "\npercentual de desconto: ";
 leia  pd ;
 sb <-  na * vha;
 td <-  ( pd / 100) * sb;
 sl <-  sb - td;
```

```
 imprima "\nsalario liquido: ",sl;
 imprima "\n";
fimprog
```

algoritmo 64

Ler uma temperatura em graus centígrados e apresentá-la convertida em graus Fahrenheit. A fórmula de conversão é: $F = \frac{9.c + 160}{5}$ *onde F é a temperatura em Fahrenheit e C é a temperatura em centígrados.*

```
prog lea37
 real  f, c;
 imprima "\ndigite o valor da temperatura em graus centigrados: ";
 leia c;
 f <-  ( 9 * c + 160)/5;
 imprima "\no valor da temperatura em graus fahrenheit e =", f;
 imprima "\n";
fimprog
```

algoritmo 65

*Calcular e apresentar o valor do volume de uma lata de óleo, utilizando a fórmula:volume = 3.14159 * R^2 * altura.*

```
prog lea38
 real  volume, altura, raio;
 imprima "\ndigite a altura da lata: ";
 leia altura;
 imprima "\ndigite o raio da lata: ";
 leia raio;
 volume <-  pi *raio ** 2 *altura;
 imprima "\no volume da lata e = ", volume;
 imprima "\n";
fimprog
```

algoritmo 66

Efetuar o cálculo da quantidade de litros de combustível gastos em uma viagem, sabendo-se que o carro faz 12 km com um litro. Deverão ser fornecidos o tempo gasto na viagem e a velocidade média.

Utilizar as seguintes fórmulas: *distância = tempo x velocidade.*
litros usados = distância / 12.

O algoritmo deverá apresentar os valores da velocidade média, tempo gasto na viagem, distância percorrida e a quantidade de litros utilizados na viagem.

```
prog lea39
 real  tempo, vel, dist, litros;
 imprima "\ndigite o tempo gasto: ";
 leia tempo;
 imprima "\ndigite  a velocidade media: ";
 leia vel;
 dist <-  tempo * vel;
 litros <-  dist / 12;
 imprima "\nvelocidade = ", vel, "\ntempo = ", tempo, "\ndistancia = ",
 dist, "\nlitros = ", litros;
 imprima "\n";
fimprog
```

algoritmo 67

Efetuar o cálculo do valor de uma prestação em atraso, utilizando a fórmula: prestação = valor + (valor(taxa/100)*tempo).*

```
prog lea40
 real  prest, valor, taxa;
 int tempo;
 imprima "\ndigite o valor da prestação: ";
 leia valor;
 imprima "\ndigite  a taxa: ";
 leia taxa;
 imprima "\ndigite o tempo(numero de meses): ";
 leia tempo;
 prest <- valor+(valor*(taxa/100)*tempo);
 imprima "\no valor da prestacao em atraso e =", prest;
 imprima "\n";
fimprog
```

algoritmo 68

Ler dois valores para as variáveis A e B, efetuar a troca dos valores de forma que a variável A passe a ter o valor da variável B e que a variável B passe a ter o valor da variável A. Apresentar os valores trocados.

```
prog lea41
 real  a, b, aux;
 imprima "\ndigite 1 numero com ponto: ";
 leia a;
 imprima "\ndigite 2 numero com ponto: ";
 leia b;
 aux <- a;
 a   <- b;
 b   <-aux;
 imprima "\na = ", a, "\nb = ", b;
```

```
 imprima "\n";
fimprog
```

algoritmo 69

Criar um algoritmo que leia o numerador e o denominador de uma fração e transformá-lo em um número decimal.

```
prog lea42
 int num, denom;
 imprima "\ndigite numerador: ";
 leia num;
 imprima "\ndigite denominador: ";
 leia denom;
 imprima "\ndecimal: ", num / denom;
 imprima "\n";
fimprog
```

algoritmo 70

Todo restaurante, embora por lei não possa obrigar o cliente a pagar, cobra 10% para o garçom. Fazer um algoritmo que leia o valor gasto com despesas realizadas em um restaurante e imprima o valor total com a gorjeta.

```
prog lea43
 real cres, cgorj;
 imprima "\nEntre com o valor da conta: ";
 leia cres;
 cgorj <- cres *1.1;
 imprima "\nO valor da conta com a gorjeta sera: ", formatar(cgorj,2);
 imprima "\n";
fimprog
```

algoritmo 71

Criar um algoritmo que leia um valor de hora e informe quantos minutos se passaram desde o início do dia.

```
prog lea44
 int hora, tminuto, minuto;
 imprima "\nentre com hora atual: ";
 leia hora;
 imprima "\nentre com minutos: ";
 leia minuto;
 tminuto <- hora * 60 + minuto;
 imprima "\nAte agora se passaram: ", tminuto, " minutos";
 imprima "\n";
fimprog
```

algoritmo 72

Criar um algoritmo que leia o valor de um depósito e o valor da taxa de juros. Calcular e imprimir o valor do rendimento e o valor total depois do rendimento.

```
prog lea45
 real deposito, taxa, valor, total;
 imprima "\nentre com depósito: ";
 leia deposito;
 imprima "\nentre coma taxa de juros: ";
 leia taxa;
 valor <- deposito*taxa/100;
 total <- deposito + valor;
 imprima "\nRendimentos: ", valor, "\nTotal: ", total;
 imprima "\n";
fimprog
```

algoritmo 73

Criar um algoritmo que receba um número real, calcular e imprimir:

- *a parte inteira do número*
- *a parte fracionária do número*
- *o número arredondado*

```
prog lea46
 real num, numfrac;
 int numi,numa;
 imprima "\nentre com um numero com parte fracionaria: ";
 leia num;
 numi <- realint((num - 0.5));
 numfrac  <- num - numi;
 numa <- realint(num + 0.00001);
 imprima "\nparte inteira: ",numi;
 imprima "\nparte fracionaria: ",formatar((numfrac + 0.00001),3);
 imprima "\nnumero arredondado: ", numa;
 imprima "\n";
fimprog
```

Qualquer dúvida, consulte o Apêndice I.

VÍDEO	
entre com um numero com parte fracionaria:7.1 parte inteira:7 parte fracionária:0.100 numero arredondado:7 entre com um numero com parte fracionaria:8.5 parte inteira:8 parte fracionária:0.500 numero arredondado:9 entre com um numero com parte fracionaria:7.49 parte inteira:7 parte fracionária:0.490 numero arredondado:7	entre com um numero com parte fracionaria:8.4999 parte inteira:8 parte fracionária:0.499 numero arredondado:8 entre com um numero com parte fracionaria:7.4999 parte inteira:7 parte fracionária:0.499 numero arredondado:7 entre com um numero com parte fracionária:8.0 parte inteira:8 parte fracionária:1.000 numero arredondado:8

algoritmo 74

Para vários tributos, a base de cálculo é o salário mínimo. Fazer um algoritmo que leia o valor do salário mínimo e o valor do salário de uma pessoa. Calcular e imprimir quantos salários mínimos ela ganha.

```
prog lea47
 real salmin, salpe, num;
 imprima "\nentre com o salario minimo: ";
 leia salmin;
 imprima "\nentre com o salario da pessoa: ";
 leia salpe;
 num <- salpe / salmin;
 imprima "\na pessoa ganha ", num, " salarios minimos";
 imprima "\n";
 fimprog
```

algoritmo 75

Criar um algoritmo que leia o peso de uma pessoa, só a parte inteira, calcular e imprimir:

- *o peso da pessoa em gramas*
- *novo peso, em gramas, se a pessoa engordar 12%*

```
prog lea48
 int peso, pesogramas, novopeso;
 imprima "\nentre com seu peso, só a parte inteira: ";
 leia peso;
 pesogramas <- peso * 1000;
 novopeso  <- pesogramas * 1.12;
 imprima "\npeso em gramas: ", pesogramas;
 imprima "\nnovo peso: ", novopeso;
 imprima "\n";
fimprog
```

algoritmo 76

Criar um algoritmo que leia um número entre 0 e 60 e imprimir o seu sucessor, sabendo que o sucessor de 60 é 0. Não pode ser utilizado nenhum comando de seleção e nem de repetição.

```
prog leia49
 int num;
 imprima "\ndigite numero: ";
 leia num;
 imprima "\nsucessor: ", (num + 1) % 61;
 imprima "\n";
fimprog
```

algoritmo 77

Ler dois números reais e imprimir o quadrado da diferença do primeiro valor pelo segundo e a diferença dos quadrados.

```
prog lea50
 real a, b, d, q;
 imprima "\ndigite 1 numero: ";
 leia a;
 imprima "\ndigite 2 numero: ";
 leia b;
 d <- (a - b)**2;
 q <-a**2 - b**2;
 imprima "\no quadrado da diferenca =",d , "\ndiferenca dos quadrados =", q;
 imprima "\n";
fimprog
```

algoritmo 78

Dado um polígono convexo de n *lados, podemos calcular o número de diagonais diferentes (nd) desse polígono pela fórmula : nd = n (n – 3) / 2. Fazer um algoritmo que leia quantos lados tem o polígono, calcule e escreva o número de diagonais diferentes (nd) do mesmo.*

```
prog lea51
 real  nd;
 int n;
 imprima "\ndigite o numero de lados do poligono: ";
 leia n;
 nd <- n * ( n - 3)  / 2;
 imprima "\nnumero de diagonais: ", nd;
 imprima "\n";
fimprog
```

algoritmo 79

Uma pessoa resolveu fazer uma aplicação em uma poupança programada. Para calcular seu rendimento, ela deverá fornecer o valor constante da aplicação mensal, a taxa e o número de meses. Sabendo-se que a fórmula usada para este cálculo é:

$$\text{valor acumulado} = P * \frac{(1+i)^n - 1}{i}$$

i = taxa
P = aplicação mensal
n= número de meses

```
prog lea52
 real  va, i, p;
 int  n;
 imprima "\ndigite o valor da aplicacao: ";
 leia p;
 imprima "\ndigite a taxa( 0 - 1): ";
 leia i;
 imprima "\ndigite o numero de meses: ";
 leia n;
 va <-p*(((1+ i )** n)-1) / i;
 imprima "\nO valor acumulado: ", va;
 imprima "\n";
fimprog
```

algoritmo 80

Criar um algoritmo que leia a quantidade de fitas que uma locadora de vídeo possui e o valor que ela cobra por cada aluguel, mostrando as informações pedidas a seguir:

- *Sabendo que um terço das fitas são alugadas por mês, exiba o faturamento anual da locadora;*

- *Quando o cliente atrasa a entrega, é cobrada uma multa de 10% sobre o valor do aluguel. Sabendo que um décimo das fitas alugadas no mês são devolvidas com atraso, calcule o valor ganho com multas por mês;*

- *Sabendo ainda que 2% de fitas se estragam ao longo do ano, e um décimo do total é comprado para reposição, exiba a quantidade de fitas que a locadora terá no final do ano.*

```
prog lea53
 int quant;
 real valAluguel, fatAnual, multas, quantFinal;
 imprima "\n Digite a quantidade de fitas: ";
 leia quant;
 imprima "\n Digite o valor do aluguel: ";
 leia valAluguel;
 fatAnual <- quant/3 * valAluguel * 12;
 imprima "\n Faturamento anual: ", fatAnual;
 multas <- valAluguel * 0.1 * (quant/3)/10;
 imprima "\n Multas mensais: ", multas;
 quantFinal <- quant – quant * 0.02 + quant/10;  /*  quant * 1.08   */
 imprima "\n Quantidade de fitas no final do ano : ", quantFinal;
 imprima "\n";
fimprog
```

algoritmo 81

Criar um algoritmo que, dado um número de conta corrente com três dígitos, retorne o seu dígito verificador, o qual é calculado da seguinte maneira:

Exemplo: número da conta: 235

- *Somar o número da conta com o seu inverso: 235+ 532 = 767*
- *multiplicar cada dígito pela sua ordem posicional e somar estes resultados: 7 6 7*

7		6		7	
X1		X2		X3	
7	+	12	+	21	=40

- *o último dígito desse resultado é o dígito verificador da conta (40 → 0).*

```
prog lea54
 int conta, inv, digito, d1, d2, d3,soma;
 imprima "\nDigite conta de tres digitos: ";
 leia conta;
 d1 <- conta div 100;
 d2 <- conta % 100 div 10;
 d3 <- conta % 100 % 10;
 inv <- d3 *100 + d2 *10 +d1;
 soma <- conta + inv;
 d1 <- (soma div 100) * 1;
 d2 <- (soma % 100 div 10) * 2;
 d3 <- (soma % 100 % 10) *3;
 digito <- (d1 +d2 +d3) % 10;
 imprima "\ndigito verificador: ", digito;
 imprima "\n";
fimprog
```

ATENÇÃO

Refazer esta lista colocando todos os testes, usando o comando **se**, quando você aprender, nos exercícios necessários:

1. testar se o divisor é diferente de 0.
2. testar se o radicando é maior ou igual a 0.
3. testar se o logaritmando é maior do que 0 e a base, maior do que 0 e a base diferente de 1.
4. testar se o seno é diferente de zero quando se deseja calcular co-tangente e co-secante.
5. testar se o co-seno é diferente de zero quando se deseja calcular tangente e secante.
6. testar se os valores para lados de figuras são maiores do que zero.
7. e outros mais.

Capítulo 3

Estruturas de seleção

CONCEITOS

Nossos algoritmos até agora seguiram um mesmo padrão: entrava-se com dados, estes eram processados e alguma informação era mostrada na tela.

Dessa forma, o computador mais parecia uma máquina de calcular. O aprendizado de novos conceitos, como a estrutura de seleção, nos dará uma visão maior da complexidade de tarefas que ele poderá executar.

Vamos refletir sobre a importância dessa estrutura, lendo com atenção as afirmativas a seguir:

1. Distribuição gratuita de cestas básicas.
2. Distribuição gratuita de cestas básicas para famílias com 4 ou mais componentes.
3. Distribuição gratuita de ingressos para o teatro, sendo dois para pessoas do sexo feminino e um para pessoas do sexo masculino.

Se observarmos essas afirmativas podemos concluir que:

- Na primeira, todas as pessoas recebem a cesta básica, o que equivaleria a um comando seqüencial.

- Na segunda, só recebem as cestas básicas as famílias com pelo menos quatro integrantes.

- Na terceira, dependendo do sexo, recebe-se um ou dois ingressos.

Assim, podemos avaliar a importância do teste nas duas últimas afirmativas, pois ações diferentes são executadas de acordo com o resultado.

Um exemplo do nosso dia-a-dia: imagine-se diante de um caixa eletrônico e suponha que sua senha seja 1234:

Na tela aparece a mensagem: O cursor (■ ou |) fica piscando:	Digite sua senha: ■	
Você digita os algarismos da sua senha	1234	1233
Neste momento, a Unidade Aritmética e Lógica (um dos componentes da **CPU**) verifica se os números que você digitou são iguais a **1234**. Caso tenham sido, aparece na tela a mensagem: **VÁLIDA**; mas se você digitou algo diferente, aparece na tela a mensagem: **INVÁLIDA**.	VÁLIDA	INVÁLIDA

Conceito é uma estrutura de controle de fluxo, executando um ou vários comandos se a condição testada for verdadeira e, em alguns casos, executando um ou vários comandos se for falsa.

SINTAXES

Seleção Simples

```
se ( condição )
{
 comando ;   ou
 < seqüência de comandos separados por ; >
}
```

A sintaxe acima representa a afirmativa **2**, pois se a família tiver, no mínimo, quatro componentes, recebe a cesta básica; mas se a família tiver menos que quatro componentes, não recebe nada.

Seleção Composta

```
se ( condição )
{
 comando ;   ou
 < seqüência de comandos separados por ; >
}
senao
{
 comando ;   ou
 < seqüência de comandos separados por ; >
}
```

A sintaxe acima representa a afirmativa 3 onde, dependendo do sexo, recebe-se um ou dois convites.

OBSERVAÇÕES

1. Podemos constatar que esta estrutura faz parte do nosso cotidiano:

- *Se eu não tiver prova, vou à praia; senão vou estudar.*
- *Se eu tiver aumento, troco de carro; senão espero o 13º salário.*
- *Se minha média for maior ou igual a sete, passo direto; senão vou à prova final.*

2. A única coisa diferente é a forma como iremos escrevê-la, pois as chaves **{** *e* **}** *são* **obrigatórias** *uma vez que delimitam os comandos que pertencem a cada bloco, assim como os parênteses* **(** *e* **)** *que delimitam a condição.*

3. Essa é uma estrutura muito importante em algoritmos porque nos dá a possibilidade de verificar o que foi digitado pelo usuário ou qual o conteúdo de uma variável após um processamento etc.

Vamos analisar cada linha da segunda sintaxe que é a mais completa, pois um ou vários comandos serão executados caso a condição testada seja verdadeira e outros comandos serão executados caso a condição seja falsa.

1ª linha: se (condição)

Condição

- *A condição é uma expressão lógica* testada pela Unidade Aritmética e Lógica, devolvendo como resposta: *verdadeiro* ou *falso*.

Convém aqui recordar os operadores relacionais e os operadores lógicos:

Operadores Relacionais	Usaremos	Operadores lógicos	Usaremos
igual	==	conjunção (e)	&&
diferente	< >	disjunção (ou)	||
maior	>	negação (não)	!
menor que	<		
maior ou igual a	>=		
menor ou igual a	<=		

a. A condição pode ser uma simples *expressão relacional* formada de *dois operandos do mesmo tipo* e de *um operador relacional* (>, <, >=, <=, == e < >).

A > B	lê-se: o conteúdo da variável A é maior do que o conteúdo da variável B?
A < 12	lê-se: o conteúdo da variável A é menor do que 12?
resp < > "S"	lê-se: o conteúdo da variável resp é diferente da letra S?
resp == "BRASIL"	lê-se: o conteúdo da variável resp é igual a BRASIL?

b. A condição pode ser uma *expressão lógica* formada de *pelo menos duas expressões relacionais* que precisarão ser unidas por um dos operadores lógicos (&& **ou** | |).

A>=1 && A < 9	lê-se: o conteúdo da variável A é maior ou igual a 1 e menor que 9?
resp == "S" | | resp == "s"	lê-se: o conteúdo da variável resp é igual a S ou igual a s?

Você deve estar perguntando: e os parênteses, não são necessários ?

Não, porque as operações aritméticas têm maior prioridade, depois as relacionais e por último as lógicas; mas, *cuidado*, nem todas as linguagens agem dessa maneira.

Você deverá ter uma atenção especial quando unir vários operadores lógicos para fazer seus testes. Veremos isso mais adiante.

2ª linha: {

Indica o início do bloco caso a condição testada seja verdadeira.

3ª linha: comando ; ou < seqüência de comandos separados por ; >

Nesta parte, são relacionados os comandos que serão executados caso a condição seja verdadeira.

4ª linha: }

Indica o fim do bloco caso a condição testada seja verdadeira.

5ª linha: senao

Este comando faz parte da estrutura do se e só deverá ser usado quando pelo menos uma ação tiver de ser executada se a condição testada for falsa.

algoritmo 83

Ler um número e, se ele for positivo, imprimir seu inverso; caso contrário, imprimir o valor absoluto do número.

Solução nº 1:

```
prog inversoabsoluto
 real numero, inverso, absoluto;
  imprima "digite numero:";
  leia numero;
  se( numero > 0. )
  { inverso <- 1 / numero;
   imprima "\ninverso: ",
   inverso;}
  senao
  { absoluto <- numero * -1;
  # ou absoluto <- abs(numero);
  imprima "\nabsoluto: ",
  absoluto;}
 imprima "\n";
fimprog
```

Solução nº 2:

```
prog inversoabsoluto
 real numero;
 imprima "digite numero:";
 leia numero;
 se( numero > 0. )
 {imprima "\ninverso: ",1 /
  numero; }
 senao
{imprima "absoluto: ", numero *
 (-1);
 #imprima "absoluto: ",
 abs(numero);
  }
 imprima "\n";
fimprog
```

VÍDEO	VÍDEO
Digite numero:5. inverso:0.2	Digite numero:5. inverso:0.2
MP Memória Principal (MP) número: 5.0 inverso: 0.2 absoluto: ???	MP Memória Principal (MP) número: 5.0

algoritmo 84

Ler um número e imprimir se ele é par ou ímpar.

```
prog parimpar
 int a;
 imprima "\nDigite numero: ";
 leia a;
 se(a % 2 == 0)
 {imprima "\nPAR";}
 senao
 {imprima "\nIMPAR";}
 imprima "\n";
fimprog
```

VÍDEO
Digite numero:24 PAR   Digite numero:25 IMPAR

Se você já entendeu tudo o que foi explicado até aqui, então será capaz de deduzir o que ficará armazenado nas variáveis do trecho do algoritmo a seguir, sabendo-se que elas tanto podem ser do tipo **int**, **real** ou **string**.

DESAFIO

```
.
.
.
se(a > b)
{ aux <- a;
  a <- b;
  b <- aux;
}
se(a > c)
{ aux <- a;
  a <- c;
  c <- aux;
}
```

```
se(b > c)
{ aux <- b;
  b <- c;
  c <- aux;
}
.
.
.
```

Resposta: Na variável *a*, ficará o menor; na variável *b*, o do meio; e, na variável *c*, o maior (se forem números); ou então ficarão as três palavras em ordem alfabética, uma vez que também é uma ordem crescente.

SES ANINHADOS (ENCAIXADOS)

Muitas vezes, em algumas aplicações, sentiremos a necessidade de tomar outras decisões dentro de uma das alternativas da estrutura do se; a isso chamamos de ses aninhados.

Vejamos um exemplo clássico de algoritmos:

algoritmo 85

```
prog descubra
 real a, b, c, max;
 imprima "\ndigite 1 numero: ";
 leia a;
 imprima "\ndigite 2 numero: ";
 leia b;
 imprima "\ndigite 3 numero: ";
 leia c;
 se (a > b)
 {
  se (a > c)
  {  max <- a ;  }
  senao
  {  max <- c;  }
 }
 senao
 {
   se ( b > c)
   {  max <- b; }
   senao
   {  max <- c ; }
 }
 imprima "\n",max;
 imprima "\n";
fimprog
```

Você descobriu o que faz esse algoritmo?

Resposta: Armazena na variável **max** o maior número entre 3, imprimindo-o. Desafio: Normalmente, o uso de **se**s aninhados melhora a performance do algoritmo. Será que nesse caso aconteceu isso? Tente usar o trecho do desafio anterior e melhore esta solução. A resposta estará mais adiante.

Outros exemplos:

algoritmo 86

Ler um número e imprimir se ele é positivo, negativo ou nulo.

```
prog pnn
 real num;
 imprima "\nDigite numero: ";
 leia num;
 se(num > 0.)
 {imprima "\nPOSITIVO";}
 senao
 { se(num < 0.)
  { imprima "\nNEGATIVO";}
  senao
  { imprima "\nNULO";}
 }
 imprima "\n";
fimprog
```

VÍDEO
Digite numero:34. POSITIVO   Digite numero:-12. NEGATIVO Digite numero:0. NULO

Você deve estar fazendo algumas perguntas:

1. *Por que não se perguntou se o número era igual a zero?*

Resposta: *Quando temos a possibilidade de três respostas, só precisamos fazer*

duas perguntas, pois a segunda pergunta nos dá as duas últimas respostas. Veja bem: se descartarmos a possibilidade de o número ser maior do que 0, ficamos com duas possibilidades: o número ser igual a 0 ou menor do que 0; dessa forma, uma pergunta é satisfatória.

2. *Mas se eu fizer, estarei errado(a)?*

Resposta: *Não, mas não é necessário.*

3. *Por que a solução não poderia ser como a a seguir?*

```
prog se
 real num;
 imprima "\nDigite numero: ";
 leia num;
 se(num > 0.)
 { imprima "\nPOSITIVO";}
 se(num < 0.)
 { imprima "\nNEGATIVO";}
 se(num == 0.)
 { imprima "\nNULO";}
 imprima "\n";
fimprog
```

Resposta: *Esta solução, embora você consiga atingir os objetivos do algoritmo, apresenta um grande inconveniente: sempre serão executados três testes, mesmo quando já tivermos classificado o número. Entretanto, na 1ª solução, outro teste só será executado se ainda não tivermos chegado a uma conclusão sobre o número.*

Esta estrutura precisa de vários ciclos para ser executada; portanto, evite usá-la desnecessariamente.

algoritmo 87

Criar um algoritmo que permita ao aluno responder qual a capital do Brasil. Todas as possibilidades deverão ser pensadas.

```
prog geo
 string resp;
 imprima "\nQual a capital do Brasil? ";
 leia resp;
 se(resp == "BRASÍLIA" || resp == "Brasília")
 {imprima "\nPARABÉNS!";}
 senao
 { se(resp=="brasília" || resp=="BRASÍLIA"|| resp=="Brazília" || resp=="brazília")
  { imprima "\nCERTO! Mas atenção para grafia: Brasília ou BRASÍLIA ";}
  senao
    { imprima "\nERRADO! ESTUDE MAIS!";}
```

```
 }
 imprima "\n";
fimprog
```

Digite no ambiente Linux por causa da acentuação.

VÍDEO

```
Qual a capital do Brasil? brasília
CERTO! Mas atenção para grafia: Brasília ou BRASÍLIA

Qual a capital do Brasil? BRASÍLIA
PARABÉNS!
Qual a capital do Brasil? Brasília
PARABÉNS!
Qual a capital do Brasil? Brazília
CERTO! Mas atenção para grafia: Brasília ou BRASÍLIA
Qual a capital do Brasil? BRAZÍLIA
CERTO! Mas atenção para grafia: Brasília ou BRASÍLIA
Qual a capital do Brasil? brazília
CERTO! Mas atenção para grafia: Brasília ou BRASÍLIA

Qual a capital do Brasil? Vitória
ERRADO! ESTUDE MAIS!
```

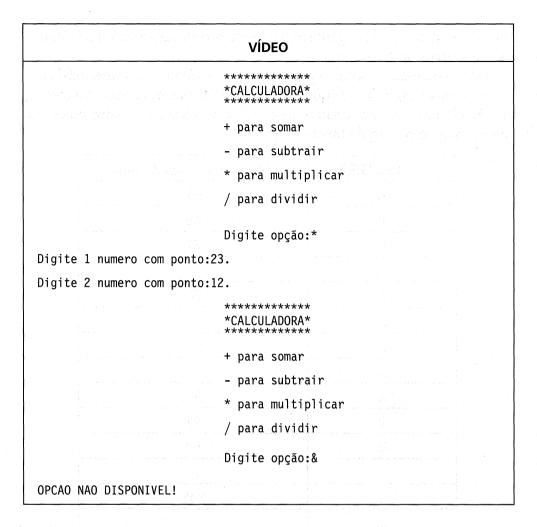
algoritmo 88

Algoritmo Calculadora

```
prog calculadora
 string resp;
 real a,b;
 imprima "\n\n\t\t\t\t*************\n";
 imprima "\t\t\t\t*CALCULADORA*\n";
 imprima "\t\t\t\t*************\n";
 imprima "\n\t\t\t\t+ para somar\n";
 imprima "\n\t\t\t\t- para subtrair\n";
 imprima "\n\t\t\t\t* para multiplicar\n";
 imprima "\n\t\t\t\t/ para dividir\n";
 imprima "\n\n\t\t\t\tDigite opcao: ";
 leia resp;
```

```
se(resp=="+")
{
 imprima "\nDigite 1 numero com ponto: ";
 leia a;
 imprima "\nDigite 2 numero com ponto: ";
 leia b;
 imprima "\nSOMA: ",a + b;
}
senao
{
  se(resp=="-")
   {
    imprima "\nDigite 1 numero com ponto: ";
    leia a;
    imprima "\nDigite 2 numero com ponto: ";
    leia b;
    imprima "\nSUBTRACAO: ",a - b;
    }
  senao
   {
   se(resp=="*")
    {
      imprima "\nDigite 1 numero com ponto: ";
      leia a;
      imprima "\nDigite 2 numero com ponto: ";
      leia b;
      imprima "\nMULTIPLICACAO: ",a * b;
      }
     senao
      {
      se(resp=="/")
       {
        imprima "\nDigite 1 numero com ponto: ";
        leia a;
        imprima "\nDigite 2 numero com ponto: ";
        leia b;
        imprima "\nDIVISAO: ",a / b;
       }
      senao
       {
        imprima "\nOPÇAO NAO DISPONIVEL!";
       }
      }
   }
}
 imprima "\n";
fimprog
```

VÍDEO

```
             *************
             *CALCULADORA*
             *************

             + para somar

             - para subtrair

             * para multiplicar

             / para dividir

             Digite opção:*

Digite 1 numero com ponto:23.

Digite 2 numero com ponto:12.

             *************
             *CALCULADORA*
             *************

             + para somar

             - para subtrair

             * para multiplicar

             / para dividir

             Digite opção:&

OPCAO NAO DISPONIVEL!
```

UM POUCO MAIS SOBRE VARIÁVEL STRING

Já vimos que a variável simples string é armazenada na memória como se fosse um vetor onde cada caractere é armazenado em uma posição conforme o esquema a seguir:

variável string

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7	byte 8	byte 9	....	byte n

Analisando sob esse prisma, podemos dizer que a variável simples string é um conjunto de n variáveis de um *único caractere* e todas com o mesmo nome.

Quando se comparam duas variáveis strings, na verdade, estamos comparando os códigos dos caracteres de cada string, baseado em seus va-

lores no código ASCII (Apêndice II). Dessa forma, não é difícil entender o uso dos sinais de >, < ou de = .

Usar o sinal de > significa perguntar se *vem depois na ordem alfabética* (ou < vem antes), uma vez que o código ASCII está em ordem alfabética, isto é, a letra A tem um código menor do que a letra B e assim sucessivamente. Veja o trecho da tabela a seguir:

CARACTERE	Código ASCII em decimal
A	65
B	66
C	67
...	...
L	76
...	...
Z	90
...	...
a	97
b	98
c	99
...	...
1	208
...	...
z	122

Observando o trecho da tabela acima, você poderá compreender por que uma palavra escrita em letras maiúsculas (ALGORITMOS) não é considerada igual a uma palavra escrita em letras minúsculas (algoritmos):

NOME1											
65	76	71	79	82	73	84	77	79	83	\0	?
0	1	2	3	4	5	6	7	8	9		

NOME2											
97	108	103	111	114	105	116	109	111	115	\0	?
0	1	2	3	4	5	6	7	8	9		

ALGUMAS QUESTÕES

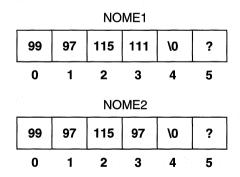
1. Algumas linguagens permitem que se monte uma expressão relacional com os nomes das variáveis:

COMPARAÇÃO
se (NOME1 > NOME2)

Outras linguagens não permitem o uso dos operadores >, < ou = e oferecem funções que viabilizam esta operação:

se (strcomp(NOME1, NOME2) **== "igual"**)	Essa expressão verifica se os conteúdos das variáveis são iguais.
se (strcomp (NOME1, NOME2) **== "maior"**)	Essa expressão verifica se o conteúdo da 1ª variável, **na ordem alfabética**, vem **depois** do conteúdo da 2ª variável.
se (strcomp (NOME1, NOME2) **== "menor"**)	Essa expressão verifica se o conteúdo da 1ª variável, **na ordem alfabética**, vem **antes** do conteúdo da 2ª variável.

No interpretador que sugerimos, você poderá fazer das duas maneiras.

2. Mas, independente do exposto acima, como a Unidade Aritmética e Lógica compara esses dados? Vejamos o exemplo a seguir, sabendo-se que a variável NOME1 armazena a palavra *caso* e a variável NOME2 armazena a palavra *casa*:

Se usássemos a expressão **Se (**NOME1 **<** NOME2**)**, a Unidade Aritmética e Lógica faria subtrações, considerando como minuendo o 1º código de NOME1 e, como subtraendo, o 1º código de NOME2 e, assim sucessivamente, até que o resultado fosse diferente de **0**.

1º passo:
99 –99 = 0
continuo porque o resultado foi **0** (“igual”).

2º passo:
97 –97 = 0
continuo porque o resultado foi **0** (“igual”).

3º passo:
115 –115 = 0
continuo porque o resultado foi **0** (“igual”).

4º passo:
111 –97 = 14
paro e vou verificar o que foi pedido

Como foi perguntado se NOME1 > NOME2 e tenho 14 (“maior”) como resposta, vou dizer que é **VERDADE.** (*Esse seria o “raciocínio” da UAL.*)

3. Algumas linguagens permitem que se atribua uma variável caracter a outra variável caracter enquanto outras oferecem uma função para produzir o mesmo efeito:

ATRIBUIÇÃO	FUNÇÃO
AUX <-NOME1;	**copia**(AUX, NOME1);

4. Algumas linguagens têm o operador // ou + (coloca o conteúdo de uma após o da outra):

CONCATENAÇÃO e ATRIBUIÇÃO	
Em algumas linguagens	Em nosso estudo
NOME <- NOME1 // NOME2;	NOME <- **strconcat**(NOME1,NOME2);

Outras linguagens não permitem nenhuma das formas acima e oferecem funções que viabilizam estas operações:

CONCATENAÇÃO e ATRIBUIÇÃO
copia(NOME, NOME1);
concatena(NOME, NOME2);

ALTERNATIVA DE MÚLTIPLAS ESCOLHAS

É uma alternativa para os ses aninhados, deixando o algoritmo com uma estrutura melhor.

Sintaxe:

```
escolha (expressão)
{
  caso  <rôtulo 1> :  comando1;
                      comando2;
                      pare;
  caso  <rôtulo 2> :  comando1;
                      comando2;
                      pare;
  caso  <rôtulo n> :  comando1;
                      comando2;
                      pare;
  senao  comando;
}
```

Considerações:

1. A expressão é avaliada e o valor será comparado com um dos rótulos.
2. A opção *senao* é opcional.
3. O rótulo será aqui definido como uma *constante caracter* (*de um caracter*) ou uma *constante numérica inteira*, embora em algumas linguagens possam ser usadas constantes caracter com mais de um caracter.
4. A estrutura é muito usada em algoritmos com menus, tornando-os mais claros do que quando usamos ses aninhados.
5. O interpretador que sugerimos nessa versão não apresenta essa estrutura.

Exemplo:
Escrever um algoritmo que leia um peso na Terra e o número de um planeta e imprima o valor do seu peso neste planeta. A relação de planetas é dada a seguir juntamente com o valor das gravidades relativas à Terra:

#	*gravidade relativa*	*Planeta*
1	0,37	Mercúrio
2	0,88	Vênus
3	0,38	Marte
4	2,64	Júpiter
5	1,15	Saturno
6	1,17	Urano

Para calcular o peso no planeta use a fórmula:

$$Pplaneta.\frac{Pterra}{10}.gravidade$$

algoritmo 89

```
prog pesodoplaneta
  int  op;
  real pterra;
  imprima  " planetas que podem ser analisados: " ;
  imprima  " 1 mercurio " ;
  imprima  " 2 venus " ;
  imprima  " 3 marte " ;
  imprima  " 4 jupiter " ;
  imprima  " 5 saturno " ;
  imprima  " 6 urano " ;
  imprima  " escolha o planeta a ser analisado " ;
  leia  op ;
  imprima  " entre com um peso na terra " ;
  leia  pterra;
  escolha ( op)
 {
  caso 1: imprima "seu peso no planeta terra é: ", (pterra/10)*0.37 ;pare;
  caso 2: imprima "seu peso no planeta terra é: ", (pterra/10)*0.88;pare;
  caso 3: imprima "seu peso no planeta terra é: ", (pterra/10)*0.38;pare;
  caso 4: imprima "seu peso no planeta terra é: ", ( pterra/10)*2.64;pare;
  caso 5: imprima "seu peso no planeta terra é: ", (pterra/10)*1.15;pare;
  caso 6: imprima "seu peso no planeta terra é: ", (pterra/10)*1.17;pare;
  senao: imprima "este planeta não pode ser analisado";
 }
 imprima "\n";
fimprog
```

EXERCÍCIOS – LISTA 2 ESTRUTURA DO SE

algoritmo 90

Entrar com um número e imprimi-lo caso seja maior que 20.

```
prog se1
 real numero;
 imprima "\n\ndigite numero: ";
 leia numero;
 se ( numero > 20. )
 {   imprima numero; }
 imprima "\n";
fimprog
```

algoritmo 91

Construir um algoritmo que leia dois valores numéricos inteiros e efetue a adição; caso o resultado seja maior que 10, apresentá-lo.

```
prog se2
 int  num1, num2, soma;
 imprima "\n\ndigite 1 numero: ";
 leia num1;
 imprima "\n\ndigite 2 numero: ";
 leia num2;
 soma <- num1 + num2;
 se ( soma > 10 )
 {   imprima "\nsoma: ", soma; }
 imprima "\n";
fimprog
```

algoritmo 92

Construir um algoritmo que leia dois números e efetue a adição. Caso o valor somado seja maior que 20, este deverá ser apresentado somando-se a ele mais 8; caso o valor somado seja menor ou igual a 20, este deverá ser apresentado subtraindo-se 5.

```
prog se3
 real  num1, num2, soma;
 imprima "\ndigite 1 numero: ";
 leia num1;
 imprima "\ndigite 2 numero: ";
 leia num2;
 soma <- num1 + num2;
 se ( soma > 20. )
 {   imprima "\nsoma: ", soma + 8; }
 senao
 { imprima "\nsoma: ", soma - 5; }
 imprima "\n";
fimprog
```

algoritmo 93

Entrar com um número e imprimir a raiz quadrada do número caso ele seja positivo e o quadrado do número caso ele seja negativo.

```
prog se4
 real numero;
 imprima "\ndigite numero: ";
 leia numero;
 se ( numero > 0. )
 {  imprima "\nraiz: ", raiz(numero); }
```

```
 senao
 {
    se ( numero < 0. )
    {   imprima "\nquadrado: ", numero ** 2; }
 }
 imprima "\n";
fimprog
```

algoritmo 94

Entrar com um número e imprimir uma das mensagens: é multiplo de 3 ou não é multiplo de 3.

```
prog se5
 int  numero;
 imprima "\ndigite numero: ";
 leia numero;
 se ( numero % 3 == 0 )
 {   imprima "\nmultiplo de 3 ";}
 senao
 {   imprima "\nnao e  multiplo de 3"; }
 imprima "\n";
fimprog
```

algoritmo 95

Entrar com um número e informar se ele é ou não divisível por 5.

```
prog se6
 int  numero;
 imprima "\ndigite numero: ";
 leia numero;
 se ( numero % 5 == 0 )
 {    imprima "\ne divisivel por 5 "; }
 senao
 {    imprima "\nnao e divisivel por 5"; }
 imprima "\n";
fimprog
```

algoritmo 96

Entrar com um número e informar se ele é divisível por 3 e por 7.

```
prog se7
 int  numero;
 imprima "\ndigite numero: ";
 leia numero;
 se (numero %3==0 && numero %7==0 ) /*poderia testar (numero %21 == 0) */
 {   imprima "\ndivisível por 3 e  por 7"; }
 senao
```

```
 { imprima "\nnão e divisível por 3 e por 7"; }
 imprima "\n";
fimprog
```

algoritmo 97

Entrar com um número e informar se ele é divisível por 10, por 5, por 2 ou se não é divisível por nenhum destes.

```
prog se8
 int  numero;
 imprima "\ndigite numero: ";
 leia numero;
 se ( numero % 10 == 0 )
 { imprima "\nmúltiplo de 10"; }
 senao
 {
  se ( numero % 2 == 0 )
  {  imprima "\nmúltiplo de 2";  }
  senao
  {
    se ( numero % 5 == 0 )
    {  imprima "\nmúltiplo de 5"; }
    senao
    {  imprima "\nnão é múltiplo de 2 nem de 5"; }
  }
 }
 imprima "\n";
fimprog
```

algoritmo 98

A prefeitura do Rio de Janeiro abriu uma linha de crédito para os funcionários estatutários. O valor máximo da prestação não poderá ultrapassar 30% do salário bruto. Fazer um algoritmo que permita entrar com o salário bruto e o valor da prestação e informar se o empréstimo pode ou não ser concedido.

```
prog se9
 real sb, vp;
 imprima "\ndigite o salario: ";
 leia sb;
 imprima "\ndigite o valor da prestacao: ";
 leia vp;
 se( vp <= 0.3 * sb )
 {imprima "\nemprestimo concedido ";}
 senao
 {imprima "\nemprestimo negado ";}
 imprima "\n";
fimprog
```

algoritmo 99

Ler um número inteiro de 3 casas decimais e imprimir se o algarismo da casa das centenas é par ou ímpar.

```
prog se10
 int num, c;
 imprima "\nnumero de 3 algarismos: ";
 leia num;
 c <- num  div  100;
 se( c % 2  == 0 )
 {imprima "\no algarismo das centenas e par: ",c;}
 senao
 {imprima "\no algarismo das centenas e  impar: ",c;}
 imprima "\n";
fimprog
```

algoritmo 100

Ler um número inteiro de 4 casas e imprimir se é ou não múltiplo de quatro o número formado pelos algarismos que estão nas casas das unidades de milhar e das centenas.

```
prog se11
 int num, c;
 imprima "\nnumero de 4 algarismos: ";
 leia num;
 c <- num  div  100;
 se( c % 4  == 0 )
 {imprima "\no numero e multiplo de 4: ",c;}
 senao
 {imprima "\no numero nao e multiplo de 4: ",c;}
 imprima "\n";
fimprog
```

algoritmo 101

Construir um algoritmo que indique se o número digitado está compreendido entre 20 e 90 ou não.

```
prog se12
 real   num;
 imprima "\ndigite numero: ";
 leia num;
 se ( num > 20. && num <  90. )
 {  imprima "\no número está na faixa de 20 a 90, exclusive"; }
 senao
 {  imprima "\no número está fora da faixa de 20 a 90";}
 imprima "\n";
fimprog
```

algoritmo 102

Entrar com um número e imprimir uma das mensagens: maior do que 20, igual a 20 ou menor do que 20.

```
prog se13
  real numero;
  imprima "\ndigite numero: ";
  leia numero;
  se ( numero > 20. )
  {  imprima "\nmaior que 20"; }
  senao
  {
   se ( numero < 20. )
   {  imprima "\nmenor que 20"; }
   senao
   { imprima "\nigual a 20"; }
  }
  imprima "\n";
fimprog
```

algoritmo 103

Entrar com o ano de nascimento de uma pessoa e o ano atual. Imprimir a idade da pessoa. Não se esqueça de verificar se o ano de nascimento é um ano válido.

```
prog se14
 int  anon, anoa;
 imprima "\nEntre com ano atual: ";
 leia anoa;
 imprima "\nEntre com ano de nascimento: ";
 leia  anon;
 se ( anon > anoa )
 {  imprima  "\nAno de Nascimento Invalido"; }
 senao
 {  imprima  "\nIdade: " , anoa - anon; }
 imprima "\n";
fimprog
```

algoritmo 104

Entrar com nome, sexo e idade de uma pessoa. Se a pessoa for do sexo feminino e tiver menos que 25 anos, imprimir nome e a mensagem: ACEITA. Caso contrário, imprimir nome e a mensagem: NÃO ACEITA. (Considerar f ou F.)

```
prog se15
 int idade;
 string nome, sexo;
 imprima "\ndigite nome: ";
```

```
 leia nome;
 imprima "\ndigite sexo: ";
 leia sexo;
 imprima "\ndigite idade: ";
 leia idade;
 se( (sexo =="F" || sexo=="f") &&  idade<25)
 {  imprima "\n",nome, " ACEITA"; }
 senao
 {  imprima  "\n",nome, " NAO ACEITA"; }
 imprima "\n";
fimprog
```

algoritmo 105

Entrar com a sigla do estado de uma pessoa e imprimir uma das mensagens:

- *carioca*
- *paulista*
- *mineiro*
- *outros estados*

```
prog se16
 string sigla;
 imprima "\ndigite sigla: ";
 leia sigla;
 se ( sigla == "RJ" ||  sigla == "rj")
 {   imprima "\ncarioca"; }
 senao
 { se (sigla =="SP" ||  sigla == "sp")
  {  imprima "\npaulista"; }
  senao
  { se ( sigla =="MG" || sigla == "mg")
    { imprima "\nmineiro";}
    senao
    { imprima  "\noutros estados"; }
   }
 }
 imprima "\n";
fimprog
```

algoritmo 106

Entrar com um nome e imprimi-lo se o primeiro caractere for a letra A (considerar letra minúscula ou maiúscula).

```
prog se17
 string nome, letra;
 imprima "\ndigite nome: ";
```

```
 leia nome;
 letra <- strprim(nome);
 se  ( letra =="A"  || letra =="a" )
 {   imprima "\n",nome; }
 imprima "\n";
fimprog
```

algoritmo 107

Entrar com o nome de uma pessoa e só imprimi-lo se o prenome for JOSÉ.

```
prog se18
 string nome;
 imprima "\ndigite nome: ";
 leia nome;
 se  (strnprim(nome, 5) == "JOSÉƀ" || nome=="JOSÉ" )
 {  imprima "\n",nome; }
 imprima "\n";
fimprog
```

ƀ significa pressionar a barra de espaço

Há duas possibilidades, pois pode ser que a pessoa só digite JOSÉ e não o nome completo.

---

algoritmo 108

Idem ao anterior, porém considerar: JOSÉ, José ou josé.

```
prog se19
 string nome, prenome;
 imprima "\ndigite nome: ";
 leia nome;
 prenome <- strnprim(nome, 5);
 se( prenome == "JOSÉƀ" ||  prenome == "Joséƀ" || prenome == "joséƀ"
 || nome=="josé" || nome =="José" || nome=="JOSÉ" ) #mesma linha
 {   imprima "\n",nome; }
 imprima "\n";
fimprog
```

algoritmo 109

Criar um algoritmo que entre com dois nomes e imprimi-los em ordem alfabética.

```
prog se20a
 string nome1,nome2;
 imprima "\ndigite 1º nome: ";
 leia nome1;
 imprima "\ndigite 2º nome: ";
 leia  nome2;
 se (nome1< nome2)
 { imprima "\n", nome1,"  ",
nome2 ; }
 senao
 { imprima  "\n", nome2,"   ",
nome1; }
 imprima "\n";
fimprog
```

```
prog se20b
 string nome1,nome2;
 imprima "\ndigite 1º nome: ";
 leia nome1;
 imprima "\ndigite 2º nome: ";
 leia  nome2;
 se (strcomp(nome1, nome2) ==
"menor")
  { imprima  "\n", nome1,"  ",
nome2; }
 senao
 { imprima "\n", nome2,"  ",
nome1; }
 imprima "\n";
fimprog
```

algoritmo 110

Criar um algoritmo que leia dois números e imprimir uma mensagem dizendo se são iguais ou diferentes.

```
prog se21
 real a, b;
 imprima "\ndigite 1º numero: ";
 leia a;
 imprima "\ndigite 2º numero: ";
 leia  b;
 se (a == b)
 { imprima "\niguais"; }
 senao
 { imprima "\ndiferentes"; }
 imprima "\n";
fimprog
```

algoritmo 111

Entrar com dois números e imprimir o maior número (suponha números diferentes).

```
prog se22
 real a, b;
 imprima "\ndigite 1º numero: ";
 leia a;
 imprima "\ndigite 2º numero: ";
```

```
 leia  b;
 se (a > b)
 {  imprima "\nmaior: ", a; }
 senao
 { imprima "\nmaior: ", b; }
 imprima "\n";
fimprog
```

algoritmo 112

Entrar com dois números e imprimir o menor número (suponha números diferentes).

```
prog se23
 real a, b;
 imprima "\ndigite 1º numero: ";
 leia a;
 imprima "\ndigite 2º numero: ";
 leia  b;
 se (a < b)
 {  imprima "\nmenor: ", a; }
 senao
 { imprima "\nmenor: ", b; }
 imprima "\n";
fimprog
```

algoritmo 113

Entrar com dois números e imprimi-los em ordem crescente (suponha números diferentes).

```
prog se24
 real a, b;
 imprima "\ndigite 1º numero: ";
 leia a;
 imprima "\ndigite 2º numero: ";
 leia  b;
 se (a < b)
 {  imprima "\n",a , "  ",b; }
 senao
 {  imprima "\n",b , "  ",a; }
 imprima "\n";
fimprog
```

algoritmo 114

Entrar com dois números e imprimi-los em ordem decrescente (suponha números diferentes).

```
prog se25
 real a, b;
```

```
 imprima "\ndigite 1º numero: ";
 leia a;
 imprima "\ndigite 2º numero: ";
 leia  b;
 se (a > b)
 {  imprima "\n",a , "  ",b; }
 senao
 { imprima "\n",b , "  ",a; }
 imprima "\n";
fimprog
```

algoritmo 115

Criar o algoritmo que deixe entrar com dois números e imprimir o quadrado do menor número e a raiz quadrada do maior número, se for possível.

```
prog se26
 real x, y;
 imprima "\ndigite valor de x: ";
 leia x;
 imprima "\ndigite valor de y: ";
 leia y;
 se(x > y)
 {imprima "\n", raiz(x), "\n", y **2;}
 senao
 { se(y > x)
  {imprima "\n", raiz(y), "\n", x **2;}
  senao
  { imprima "\nnumeros iguais";}
 }
 imprima "\n";
fimprog
```

algoritmo 116

Entrar com três números e imprimir o maior número (suponha números diferentes).

```
prog se27solucao1
 real a, b, c;
 imprima "\ndigite 1 numero: ";
 leia a;
 imprima "\ndigite 2 numero: ";
 leia b;
 imprima "\ndigite 3 numero: ";
 leia c;
 se (a > b)
 {
   se (a > c)
   { imprima  "\nmaior: ", a ; }
```

```
    senao
    { imprima "\nmaior: ", c; }
  }
  senao
  {
    se ( b > c)
    { imprima "\nmaior: ", b; }
    senao
    { imprima "\nmaior: ", c ; }
  }
  imprima "\n";
fimprog
```

```
prog se27solucao2
 real a, b, c;
 imprima "\ndigite 1 numero: ";
 leia a;
 imprima "\ndigite 2 numero: ";
 leia b;
 imprima "\ndigite 3 numero: ";
 leia c;
 se (   a > b  &&   a > c )
 { imprima  "\nmaior: ", a ; }
 senao
 {
    se  ( b > c)
    { imprima  "\nmaior: ", b ; }
    senao
    { imprima  "\nmaior: ", c ; }
  }
 imprima "\n";
fimprog
```

algoritmo 117

Entrar com três números e armazenar o maior número na variável de nome maior (suponha números diferentes).

Esta é uma solução alternativa e melhor para o algoritmo anterior.

```
prog se28solucao1
 real a, b, c, maior;
 imprima "\ndigite 1 numero: ";
 leia a;
 imprima "\ndigite 2 numero: ";
 leia b;
 imprima "\ndigite 3 numero: ";
 leia c;
```

```
 se (a > b)
 {  maior <- a; }
 senao
 {  maior <- b ; }
 se (c > maior)
 {   maior <- c; }
 imprima "\nmaior: ", maior;
 imprima "\n";
fimprog

prog se28solucao2
 real a, b, c, maior;
 imprima "\ndigite 1 numero: ";
 leia a;
 imprima "\ndigite 2 numero: ";
 leia b;
 imprima "\ndigite 3 numero: ";
 leia c;
 maior <- a;
 se (b > maior)
 {  maior <- b ; }
 se (c > maior)
 {  maior <- c; }
 imprima "\nmaior: ", maior;
 imprima "\n";
fimprog
```

algoritmo 118

Entrar com três números e imprimi-los em ordem crescente (suponha números diferentes).

↪ *Várias são as soluções para colocar três números em ordem crescente, decrescente, maior e menor ou intermediário. Acreditamos que esta solução seja bem simples e tem como objetivo armazenar na variável a o menor valor; na variável b, o segundo valor; e na variável c, o maior valor.*

```
prog  se29
 real a, b, c, aux;
 imprima "\ndigite 1 numero: ";
 leia a;
 imprima "\ndigite 2 numero: ";
 leia b;
 imprima "\ndigite 3 numero: ";
 leia c;
 se ( a  > b )
 { aux <- a; a <- b; b <- aux; }
```

```
 se ( a  > c )
 { aux <- a; a <- c; c <- aux; }
 se ( b > c )
 { aux <- b; b <- c; c <- aux; }
 imprima "\nOrdem Cresecnte: ", a, "  ",b, "  ",c ;
 imprima "\n";
fimprog
```

algoritmo 119

Entrar com três números e imprimi-los em ordem decrescente (suponha números diferentes).

```
prog  se30
 real a, b, c, aux;
 imprima "\ndigite 1 numero: ";
 leia a;
 imprima "\ndigite 2 numero: ";
 leia b;
 imprima "\ndigite 3 numero: ";
 leia c;
 se ( a  > b )
 { aux <- a; a <- b; b <- aux; }
 se ( a  > c )
 { aux <- a; a <- c; c <- aux; }
 se ( b > c )
 { aux <- b; b <- c; c <- aux; }
 imprima "\nOrdem Decresecente: ", c, "  ",b, "  ",a ;
 imprima "\n";
fimprog
```

algoritmo 120

Entrar com três números e armazená-los em três variáveis com os seguintes nomes: maior, intermediário e menor (suponha números diferentes).

```
prog se31a
 real a, b, c, maior, intermediario, menor;
 imprima "\ndigite 1 numero: ";
 leia a;
 imprima "\ndigite 2 numero: ";
 leia b;
 imprima "\ndigite 3 numero: ";
 leia c;
 se ( a > b)
 { se  ( c > a)
   {  maior <- c;
      intermediario <- a ;
      menor <- b;
```

```
  }
  senao
  { se  ( c > b)
    { maior <- a;
      intermediario <- c;
      menor <- b;
    }
    senao
    {  maior <- a;
       intermediario <- b;
       menor <- c ;
    }
  }
 }
 senao
 { se  ( c >b)
   {  maior <- c;
      intermediario <- b;
      menor <- a;
  }
  senao
  { se  ( c > a)
    { maior <- b;
      intermediario <- c;
      menor <- a;
    }
    senao
    {   maior <- b;
        intermediario <- a;
         menor <-c;
    }
  }
 }
 imprima "\nmaior: ", maior;
 imprima "\nintermediario: ",intermediario;
 imprima "\nmenor: ", menor;
 imprima "\n";
fimprog

prog  se31b
 real a, b, c, aux, maior, intermediario,menor;
 imprima "\ndigite 1 numero: ";
 leia a;
 imprima "\ndigite 2 numero: ";
 leia b;
 imprima "\ndigite 3 numero: ";
 leia c;
 se ( a  > b )
 {    aux <- a;       a <- b;       b <- aux; }
```

```
se ( a  > c )
{     aux <- a;      a <- c;      c <- aux; }
se ( b > c )
{     aux <- b;      b <- c;      c <- aux; }
maior <- c;
intermediario <- b;
menor <- a;
imprima "\nmaior: ", maior;
imprima "\nintermediario: ",intermediario;
imprima "\nmenor: ",menor;
imprima "\n";
fimprog
```

algoritmo 121

Efetuar a leitura de cinco números inteiros diferentes e identificar o maior e o menor valor.

```
prog se32
 int  n1, n2, n3, n4, n5, maior, menor;
 imprima "\ndigite 1 numero: ";
 leia n1;
 imprima "\ndigite 2 numero: ";
 leia n2;
 imprima "\ndigite 3 numero: ";
 leia n3;
 imprima "\ndigite 4 numero: ";
 leia n4;
 imprima "\ndigite 5 numero: ";
 leia n5;
 se (n1< >n2 && n1< >n3 && n1< >n4 && n1< >n5 &&  n2< >n3 && n2< >n4
 && n2< >n5  && n3< >n4 && n3<  >n5 && n4< >n5)
 {
 /* Teste necessário porque o exercício fala sobre os números serem
 diferentes. Veja como é importante saber COMBINAÇÃO */
 se (n1 > n2)
 {  maior <- n1;    menor <- n2; }
 senao
 {  maior <- n2;   menor <- n1; }
 se (n3 > maior)
 { maior <- n3; }
 senao
 {
   se (n3 < menor)
   {  menor <- n3;  }
 }
 se (n4 > maior)
 { maior <- n4; }
```

```
 senao
 {
   se (n4 < menor)
   { menor <- n4;  }
 }
 se (n5 > maior)
 {  maior <- n5; }
 senao
 {
   se (n5 < menor)
   {  menor <- n5;  }
 }
 imprima "\nMaior= ", maior;
 imprima "\nMenor= ", menor;
 }
 senao
 { imprima "\nOs valores devem ser diferentes ";}
 imprima "\n";
fimprog
```

Um algoritmo para achar o menor ou o maior com vários números fica extremamente longo. Quando você aprender as estruturas de repetição, verá que esse algoritmo poderá ser melhorado.

algoritmo 122

Ler três números e imprimir se eles podem ou não ser lados de um triângulo.

```
prog se33
 real a, b, c;
 imprima "\ndigite 1 lado: ";
 leia a;
 imprima "\ndigite 2 lado: ";
 leia b;
 imprima "\ndigite 3 lado: ";
 leia c;
 se ( a < b + c  &&   b < a + c &&  c < a + b )
 {  imprima  "\nPodem ser lados de um triangulo"; }
 senao
 { imprima "\nNão podem ser lados de um triangulo"; }
 imprima "\n";
fimprog
```

algoritmo 123

Ler três números, os possíveis lados de um triângulo, e imprimir a classificação segundo os lados.

```
prog se34
 real a, b, c;
 imprima "\ndigite 1 lado: ";
 leia a;
 imprima "\ndigite 2 lado: ";
 leia b;
 imprima "\ndigite 3 lado: ";
 leia c;
 se ( a < b + c  &&  b < a + c &&  c < a + b )
 {
   se  ( a == b  &&  a == c)
   { imprima "\nTriangulo equilatero";  }
   senao
   {
      se ( a == b  ||  a == c ||  b == c)
      { imprima "\nTriangulo isosceles";  }
     senao
     { imprima "\nTriangulo escaleno";  }
   }
 }
 senao
 {   imprima "\nAs medidas não formam  um triangulo"; }
 imprima "\n";
fimprog
```

algoritmo 124

Ler três números, os possíveis lados de um triângulo, e imprimir a classificação segundo os ângulos.

```
prog se35
 real a, b, c, maior, lados;
 imprima "\ndigite 1 lado: ";
 leia a;
 imprima "\ndigite 2 lado: ";
 leia b;
 imprima "\ndigite 3 lado: ";
 leia c;
 se ( a < b + c &&  b < a + c &&   c < a + b )
 {
    se ( a > b  &&    a > c )
    {   maior <- a ** 2;   lados  <- b ** 2 + c ** 2;   }
    senao
    { se  ( b > c)
      { maior <- b ** 2; lados  <- a ** 2 + c ** 2; }
      senao
      { maior <- c ** 2; lados  <- a **2 + b ** 2 ; }
     }
```

```
    se ( maior == lados )
    { imprima "\nTriangulo Retangulo"; }
    senao
    { se ( maior > lados )
      { imprima "\nTriangulo Obtusangulo"; }
      senao
      { imprima "\nTriangulo Acutangulo"; }
    }
 }
 senao
 { imprima "\nas medidas não formam  um  triangulo"; }
 imprima "\n";
fimprog
```

algoritmo 125

Entrar com a idade de uma pessoa e informar:

- *se é maior de idade*
- *se é menor de idade*
- *se é maior de 65 anos*

```
prog se36
 int   idade;
 imprima "\ndigite sua idade: ";
 leia idade;
 se( idade >= 65 )
 { imprima "\nmaior de 65"; }
 senao
 {  se ( idade >= 18 )
   { imprima "\nmaior de idade"; }
   senao
   {  imprima "\nmenor de idade"; }
 }
 imprima "\n";
fimprog
```

algoritmo 126

Ler um número e imprimir se ele é igual a 5, a 200, a 400, se está no intervalo entre 500 e 1000, inclusive, ou se ele está fora dos escopos anteriores.

```
prog se37
 real x;
 imprima "\ndigite valor de X: ";
 leia  x ;
 se ( x == 5. )
 { imprima  "\nO numero é o 5"; }
```

```
 senao
 { se ( x ==200. )
  { imprima  "\nO numero é o 200"; }
  senao
  { se ( x == 400. )
    { imprima  "O numero é o 400"; }
    senao
    { se ( x >= 500. &&  x <= 1000.)
      { imprima "\nIntervalo 500-1000"; }
      senao
      { imprima  "\nFora do  escopo"; }
    }
  }
 }
 imprima "\n";
fimprog
```

algoritmo 127

Entrar com nome, nota da PR1 e nota da PR2 de um aluno. Imprimir nome, nota da PR1, nota da PR2, média e uma das mensagens: Aprovado, Reprovado ou em Prova Final (a média é 7 para aprovação, menor que 3 para reprovação e as demais em prova final).

```
prog se38
 real nota1, nota2, media;
 string nome;
 imprima "\ndigite nome: ";
 leia nome;
 imprima "\ndigite 1ª nota: ";
 leia nota1;
 imprima "\ndigite 2ª nota: ";
 leia nota2;
 media <-  (nota1 + nota2)/ 2;
 se ( media >=7. )
 {  imprima nome,"   ", media, " AP"; }
 senao
 { se ( media < 3. )
   { imprima nome,"   ", media, " RP"; }
   senao
   { imprima  nome,"   ",  media, " PF"; }
  }
 imprima "\n";
fimprog
```

algoritmo 128

Entrar com um verbo no infinitivo e imprimir uma das mensagens:

- *verbo não está no infinitivo*
- *verbo da 1ª conjugação*
- *verbo da 2ª conjugação*
- *verbo da 3ª conjugação*

```
prog se39
 string verbo, letra;
 int  n;
 imprima "\ndigite verbo: ";
 leia verbo;
 letra <- strult(verbo);
 se ( letra == "R" || letra =="r")
 {
   n <- strtam(verbo) - 2;
   letra <- strelem(verbo,n);
  se  ( letra =="A"  || letra == "a" )
  { imprima "\n1ª conjugacao"; }
  senao
  {
    se ( letra =="E" || letra =="e"  || letra =="o" ||  letra =="O")
    {  imprima "\n2ª conjugacao"; }
    senao
    {
       se  ( letra =="I"  || letra == "i" )
       { imprima  "\n3ª conjugacao"; }
       senao
       { imprima "\nnao existe verbo com terminacao ur"; }
    }
  }
 }
 senao
 { imprima "\nnao e  um verbo no infinitivo"; }
 imprima "\n";
fimprog
```

algoritmo 129

Entrar com o salário de uma pessoa e imprimir o desconto do INSS segundo a tabela a seguir:

menor ou igual a R$ 600,00	*isento*
maior que R$ 600,00 e menor ou igual a R$ 1200,00	*20%*
maior que R$ 1200,00 e menor ou igual a R$2000,00	*25%*
maior que R$ 2000,00	*30%*

```
prog se40
real salario, desconto;
 imprima "\ndigite salario: ";
 leia salario;
 se ( salario <= 600. )
 {  desconto <- 0.;}
 senao
 { se  ( salario <= 1200. )
 {  desconto <- salario * 0.2; }
 senao
 { se  ( salario <= 2000. )
   {  desconto <- salario * 0.25; }
   senao
   {  desconto <- salario * 0.30;}
  }
 }
 imprima "\ndesconto: ",desconto;
 imprima "\n";
fimprog
```

algoritmo 130

Um comerciante comprou um produto e quer vendê-lo com um lucro de 45% se o valor da compra for menor que R$ 20,00; caso contrário, o lucro será de 30%. Entrar com o valor do produto e imprimir o valor da venda.

```
prog se41
 real valor;
 imprima "\ndigite valor do produto: ";
 leia valor;
 se( valor < 20. )
 {  imprima  "\nValor de venda: ", valor * 1.45 ; }
 senao
 { imprima  "\nValor de venda: " , valor *1.3; }
 imprima "\n";
fimprog
```

algoritmo 131

A turma de Programação I, por ter muitos alunos, será dividida em dias de provas. Após um estudo feito pelo coordenador, decidiu-se dividi-la em três grupos. Fazer um algoritmo que leia o nome do aluno e indicar a sala em que ele deverá fazer as provas, tendo em vista a tabela a seguir e sabendo-se que todas as salas se encontram no bloco F:

A – K : sala 101

L – N : sala 102

O – Z : sala 103

```
prog se42
 string nome,L;
 imprima "\nDigite nome: ";
 leia nome;
 L<-strprim(nome);
 se  ((L>="A" && L<="K") ||  (L>="a"  && L<="k"))
 {  imprima "\n", nome, " sala: 101"; }
 senao
 {
  se ((L>="L" && L<="N") ||  (L>="l" && L<="n"))
  {  imprima "\n", nome, " sala: 102"; }
  senao
  {
    se ( (L>="O" && L<="Z") ||  (L>="o"  && L<="z"))
    {   imprima "\n", nome, " sala: 103";  }
    senao
    {  imprima "\n NOME INVALIDO"; }
   }
  }
  imprima "\n";
fimprog
```

algoritmo 132

Fazer um algoritmo que possa converter uma determinada quantia dada em reais para uma das seguintes moedas:

- *f – franco suíço*
- *l – libra esterlina*
- *d – dólar*
- *m – marco alemão*

```
prog se43
 real r,l, d,f, m;
 string moeda;
 imprima "\ndigite valor em reais a ser convertido: ";
 leia r;
 imprima "\ndigite f - franco suico l - libra esterlina d - dolar m -
marco alemao: ";
 leia moeda;
 se( moeda =="f" || moeda =="F")
 { imprima "\ndigite valor de um franco suico em reais: ";
  leia f;
  imprima "\nTotal de francos suicos: ", r /f;}
 senao
 { se( moeda =="l" || moeda =="L")
  { imprima "\ndigite valor de uma libra esterlina em reais: ";
    leia l;
```

```
    imprima "\nTotal de libras esterlinas: ", r /l;}
  senao
  { se( moeda =="d" || moeda =="D")
    { imprima "\ndigite valor de um dolar em reais: ";
      leia d;
      imprima "\nTotal de dolares: ", r /d;}
    senao
    { se( moeda =="m" || moeda =="M")
      { imprima "\ndigite valor de um marco alemao em reais: ";
        leia m;
        imprima "\nTotal de marcos alemaes: ", r /m;}
     senao
     { imprima "\nmoeda desconhecida";}
    }
  }
 }
 imprima "\n";
fimprog
```

algoritmo 133

Segundo uma tabela médica, o peso ideal está relacionado com a altura e o sexo. Fazer um algoritmo que receba a altura e o sexo de uma pessoa, calcular e imprimir o seu peso ideal, utilizando as seguintes fórmulas:

- *para homens:* *(72.7 * H) – 58*
- *para mulheres:* *(62.1 * H) – 44.7*

```
prog se44
 string sexo;
 real h, peso;
 imprima "\nentre com a altura: ";
 leia  h;
 imprima "\nentre com o sexo M / F: ";
 leia sexo;
 se( sexo == "M" || sexo == "m")
 {peso <-  72.7 * h  - 58;}
 senao
 {peso <- 62.1 * h  - 44.7;}
 imprima "\nseu peso ideal : ", peso;
 imprima "\n";
fimprog
```

algoritmo 134

A confederação brasileira de natação irá promover eliminatórias para o próximo mundial. Fazer um algoritmo que receba a idade de um nadador e imprimir a sua categoria segundo a tabela a seguir:

Categoria	*Idade*
Infantil A	*5 – 7 anos*
Infantil B	*8 – 10 anos*
Juvenil A	*11 – 13 anos*
Juvenil B	*14 – 17 anos*
Sênior	*maiores de 18 anos*

```
prog se45
 int idade;
 imprima "\nentre com a idade: ";
 leia idade;
 se( idade < 5 )
 { imprima "\nnao existe categoria para essa idade" ;}
 senao
 { se( idade <= 7 )
   { imprima "\ncategoria Infantil A" ;}
   senao
   { se( idade <= 10 )
     { imprima "\ncategoria Infantil B";}
     senao
     { se( idade <= 13 )
       { imprima "\ncategoria Juvenil A"; }
       senao
       { se( idade <= 17 )
         { imprima "\ncategoria Juvenil B";}
        senao
        { imprima "\ncategoria Senior";}
       }
     }
   }
 }
 imprima "\n";
fimprog
```

algoritmo 135

Criar um algoritmo que leia a idade de uma pessoa e informar a sua classe eleitoral:

- *não-eleitor (abaixo de 16 anos)*
- *eleitor obrigatório (entre 18 e 65 anos)*
- *eleitor facultativo (entre 16 e 18 anos e maior de 65 anos)*

```
prog se46
 int idade;
 imprima "\ndigite idade: ";
```

```
 leia idade;
 se ( idade < 16 )
 {  imprima "\nNao eleitor"; }
 senao
 {
   se ( idade > 65 )
   { imprima "\nEleitor facultativo"; }
   senao
   { imprima "\nEleitor obrigatorio"; }
 }
 imprima "\n";
fimprog
```

algoritmo 136

Depois da liberação do governo para as mensalidades dos planos de saúde, as pessoas começaram a fazer pesquisas para descobrir um bom plano, não muito caro. Um vendedor de um plano de saúde apresentou a tabela a seguir. Criar um algoritmo que entre com o nome e a idade de uma pessoa e imprimir o nome e o valor que ela deverá pagar.

- *até 10 anos – R$ 30,00*
- *acima de 10 até 29 anos - R$ 60,00*
- *acima de 29 até 45 anos - R$ 120,00*
- *acima de 45 até 59 anos - R$ 150,00*
- *acima de 59 até 65 anos - R$ 250,00*
- *maior que 65 anos – R$ 400,00*

```
prog se47
 int  idade;
 string nome;
 imprima "\nEntre com nome: ";
 leia  nome;
 imprima "\nEntre com a idade: ";
 leia idade;
 se ( idade <= 10 )
 {  imprima "\n",nome," pagara R$ 30,00"; }
 senao
 {
   se ( idade <= 29 )
   { imprima  "\n", nome, "  pagara R$ 60,00"; }
   senao
   {
     se ( idade <= 45 )
     {  imprima  "\n", nome, "  pagara R$ 120,00"; }
     senao
```

```
    {
      se ( idade <= 59 )
      { imprima  "\n", nome, "  pagara R$ 150,00"; }
      senao
      {
        se ( idade <= 65 )
        {  imprima  "\n", nome, "  pagara R$ 250,00";  }
        senao
        {  imprima  "\n", nome, "  pagara R$ 400,00";  }
      }
    }
  }
 }
 imprima "\n";
fimprog
```

algoritmo 137

Ler três valores inteiros (variáveis a, b e c) e efetuar o cálculo da equação de segundo grau, apresentando: as duas raízes, *se para os valores informados for possível fazer o cálculo (delta positivo ou zero); a mensagem* "Não há raízes reais", *se não for possível fazer o cálculo (delta negativo); e a mensagem* "Não é equação do segundo grau", *se o valor de a for igual a zero.*

```
prog se48
 real a, b, c, d, x1, x2;
 imprima "\nEntre com valor de a: ";
 leia a;
 imprima "\nEntre com valor de b: ";
 leia  b;
 imprima "\nEntre com valor de c: ";
 leia  c;
 se (a == 0.)
 {  imprima "\n Nao e equacao do 2 grau";}
 senao
 {
  d<- b**2 - 4 * a * c;
  se (d >=0.)
  {
    d<- raiz(d);
    x1<- (-b + d)/ (2 *a);
    x2 <-(-b - d)/ (2 *a);
    imprima "\n X1= ", x1, "\t\tX2= ", x2;
  }
  senao
  {  imprima "\nNao há raizes reais"; }
 }
 imprima "\n";
fimprog
```

algoritmo 138

Ler um número inteiro entre 1 e 12 e escrever o mês correspondente. Caso o usuário digite um número fora desse intervalo, deverá aparecer uma mensagem informando que não existe mês com este número.

```
prog se49
 int  num;
 imprima "\ndigite vumnumero de 1 - 12: ";
 leia  num ;
 se (num==1)
 { imprima "\njaneiro"; }
 senao
 {  se (num==2)
    {  imprima "\nfevereiro";  }
    senao
    { se (num==3)
      {  imprima "\nmarco";  }
     senao
     { se (num==4)
      { imprima "\nabril"; }
      senao
      { se (num==5)
        { imprima "\nmaio"; }
        senao
        { se (num==6)
          { imprima "\njunho"; }
          senao
          { se (num==7)
            {  imprima "\njulho";  }
            senao
            { se (num==8)
             { imprima "\nagosto"; }
             senao
             { se (num==9)
               { imprima "\nsetembro"; }
               senao
               { se (num==10)
                 { imprima "\noutubro"; }
                 senao
                 {  se (num==11)
                   {  imprima "\nnovembro"; }
                   senao
                   { se (num==12)
                     { imprima "\ndezembro";  }
                     senao
                     {  imprima "\nnao existe mes correspondente"; }
```

```
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
 imprima "\n";
fimprog

prog se49comescolha
 # nao disponível nesta versao
 int  num;
 imprima "\ndigite vumnumero de 1 - 12: ";
 leia  num ;
 escolha (num)
 {
   caso 1: imprima "\njaneiro"; pare;
   caso 2: imprima "\nfevereiro"; pare;
   caso 3:imprima "\nmarço"; pare;
   caso 4: imprima "\nabril"; pare;
   caso 5: imprima "\nmaio";pare;
   caso 6: imprima "\njunho"; pare;
   caso 7: imprima "\njulho"; pare;
   caso 8: imprima "\nagosto"; pare;
   caso 9: imprima "\nsetembro";pare;
   caso 10: imprima "\noutubro"; pare;
   caso 11: imprima "\nnovembro"; pare;
   caso12: imprima "\ndezembro"; pare;
   senao:  imprima "\nnao existe mes correspondente a este numero";
 }
 imprima "\n";
fimprog
```

algoritmo 139

Sabendo que somente os municípios que possuem mais de 20.000 eleitores aptos têm segundo turno nas eleições para prefeito caso o primeiro colocado não tenha mais do que 50% dos votos, fazer um algoritmo que leia o nome do município, a quantidade de eleitores aptos, o número de votos do candidato mais votado e informar se ele terá ou não segundo turno em sua eleição municipal.

```
prog se50
 int ne, votos;
```

```
 string nome;
 imprima "\nDigite nome do Municipio: ";
 leia nome;
 imprima "\nnumero de eleitores aptos: ";
 leia ne;
 imprima "\nnumero de votos do candidato mais votado: ";
 leia votos;
 se (ne >20000  &&  votos <= ne div 2)
 { imprima "\n",nome," tera segundo turno"; }
 senao
 { imprima "\n",nome," nao terá segundo turno"; }
 imprima "\n";
fimprog
```

algoritmo 140

Um restaurante faz uma promoção semanal de descontos para clientes de acordo com as iniciais do nome da pessoa. Criar um algoritmo que leia o primeiro nome do cliente, o valor de sua conta e se o nome iniciar com as letras A, D, M ou S, dar um desconto de 30%. Para o cliente cujo nome não se inicia por nenhuma dessas letras, exibir a mensagem "Que pena. Nesta semana o desconto não é para seu nome; mas continue nos prestigiando que sua vez chegará".

```
prog se51
 string nome,L;
 real vc, vcd;
 imprima "\nDigite nome: ";
 leia nome;
 imprima "\nDigite valor da conta: ";
 leia vc;
 L<-strprim(nome);
 se( L=="A" || L=="a" || L=="D" ||  L=="d" || L=="M" || L=="m" || L=="S"
 || L=="s" )
 { imprima "\n", nome, " valor da conta com desconto: R$ ", formatar(vc *
 0.7, 2); }
 senao
 { imprima "\n Que pena. Nesta semana o desconto não é para seu nome; mas
 continue nos prestigiando que sua vez chegará";  }
 imprima "\n";
fimprog
```

algoritmo 141

Criar um algoritmo que leia o nome e o total de pontos de três finalistas de um campeonato de pingue-pongue e exibir a colocação da seguinte forma:

Vencedor:____________________ XXXX ptos
Segundo colocado:_______________ XXXX ptos
Terceiro colocado:_______________ XXXX ptos

```
prog  se52
 int p1, p2, p3, aux;
 string n1,n2,n3,auxn;
 imprima "\ndigite 1 nome: ";
 leia n1;
 imprima "\ndigite pontos: ";
 leia p1;
 imprima "\ndigite 2 nome: ";
 leia n2;
 imprima "\ndigite pontos: ";
 leia p2;
 imprima "\ndigite 3 nome: ";
 leia n3;
 imprima "\ndigite pontos: ";
 leia p3;
 se ( p1  < p2 )
 { aux <- p1; p1 <- p2; p2 <- aux; auxn <- n1; n1 <- n2; n2 <- auxn; }
 se ( p1  < p3 )
 { aux <- p1; p1 <- p3; p3 <- aux; auxn <- n1; n1 <- n3; n3 <- auxn; }
  se ( p2 < p3 )
 { aux <- p2; p2 <- p3; p3 <- aux; auxn <- n2; n2 <- n3; n3 <- auxn; }
 imprima "\nVENCEDOR         : ", n1, "  ",p1, " pontos";
 imprima "\nSEGUNDO COLOCADO : ", n2, "  ",p2, " pontos";
 imprima "\nTERCEIRO COLOCADO: ", n3, "  ",p3, " pontos";
 imprima "\n";
fimprog
```

algoritmo 142

Em um campeonato nacional de arco-e-flecha, tem-se equipes de três jogadores para cada estado. Sabendo-se que os arqueiros de uma equipe não obtiveram o mesmo número de pontos, criar um algoritmo que informe se uma equipe foi classificada, de acordo com a seguinte especificação:

- *ler os pontos obtidos por cada jogador da equipe;*
- *mostrar esses valores em ordem decrescente;*
- *se a soma dos pontos for maior do que 100, imprimir a média aritmética entre eles; senão, imprimir a mensagem "Equipe desclassificada".*

```
prog se53
 int p1, p2, p3, aux, soma;
 imprima "\n Digite os pontos do primeiro jogador: ";
 leia p1;
 imprima "\n Digite os pontos do segundo jogador: ";
 leia p2;
```

```
 imprima "\n Digite os pontos do terceiro jogador: ";
 leia p3;
 se(p1 > p2)
 { aux<- p1 ; p1<- p2; p2<- aux; }
 se(p1 > p3)
 { aux<- p1 ; p1<- p3; p3<- aux; }
 se(p2 > p3)
 { aux<- p2 ; p2<- p3; p3<- aux; }
 imprima "\n p1=", p1, "\n p2=", p2, "\n p3=", p3;
 soma <-  p1 + p2 + p3;
 se(soma > 100)
 { imprima "\nMedia=", formatar(soma/3,2);}
 senao
 { imprima "\nEquipe desclassificada!";}
 imprima "\n";
fimprog
```

algoritmo 143

Criar um algoritmo que verifique a(s) letra(s) central(is) de uma palavra. Se o número de caracteres for ímpar, ele verifica se a letra central é uma vogal; caso contrário, verifica se é um dos dígrafos rr *ou* ss *(só precisa testar letras minúsculas).*

```
prog  se54
 string pal, p1, p2, p3;
 int n1, n2;
 imprima "\ndigite uma palavra: ";
 leia pal;
 se(strtam(pal) % 2 == 0)
 { n1<-strtam(pal) div 2;
   n2<- n1-1;
   p1<-strelem(pal,n1);
   p2<-strelem(pal,n2);
   p3<-strconcat(p2,p1);
   se( p3  ==  "rr" || p3  ==  "ss" )
   {imprima "\nE rr OU ss";}
   senao
   {imprima "\nNAO E rr OU ss";}
 }
 senao
 { n1<-strtam(pal) div 2;
   p1<-strelem(pal,n1);
   se( p1 == "a" || p1 == "e"|| p1 == "i"  || p1 == "o" || p1 == "u")
   {imprima "\nE VOGAL";}
   senao
   {imprima "\nNAO E VOGAL";}
 }
 imprima "\n";
fimprog
```

algoritmo 144

O banco XXX concederá um crédito especial com juros de 2% aos seus clientes de acordo com o saldo médio no último ano. Fazer um algoritmo que leia o saldo médio de um cliente e calcule o valor do crédito de acordo com a tabela a seguir. Imprimir uma mensagem informando o saldo médio e o valor do crédito.

SALDO MÉDIO	*PERCENTUAL*
de 0 a 500	*nenhum crédito*
de 501 a 1000	*30% do valor do saldo médio*
de 1001 a 3000	*40% do valor do saldo médio*
acima de 3001	*50% do valor do saldo médio*

```
prog se55
 real  saldomedio, credito;
 imprima "\ndigite saldo medio: ";
 leia  saldomedio;
 se( saldomedio < 501. )
 {  credito <-0.; }
 senao
 {  se(saldomedio < 1001.)
    {  credito <- saldomedio  *  0.3;}
      senao
    { se(saldomedio < 3001.)
      { credito <- saldomedio  *  0.4;}
        senao
      { credito <- saldomedio  *  0.5;}
    }
  }
 se(credito < > 0.)
 { imprima "\nComo seu saldo era de:ƀ",saldomedio,"ƀseu credito sera
            de:ƀ",credito;}# continuacao da linha anterior
 senao
 { imprima "\nComo seu saldo era de:ƀ", saldomedio,",ƀvoce nao tera
            nenhum credito";}# continuacao da linha anterior
 imprima "\n";
fimprog
```

algoritmo 145

A biblioteca de uma universidade deseja fazer um algoritmo que leia o nome do livro que será emprestado, o tipo de usuário (professor ou aluno) e possa imprimir um recibo conforme mostrado a seguir. Considerar que o professor tem dez dias para devolver o livro e o aluno só três dias.

Nome do livro
Tipo de usuário:
Total de Dias:

```
prog se56
 string livro, tipo;
 int  dia;
 imprima "\ndigite nome do livro: ";
 leia  livro;
 imprima "\ndigite: professor / aluno: ";
 leia  tipo;
 se( tipo == "professor" ||  tipo == "Professor" ||  tipo == "PROFESSOR")
 {  dia <-10 ;}
 senao
 { se( tipo == "aluno" ||  tipo == "Aluno" ||  tipo == "ALUNO")
   {  dia <-3 ;}
   senao
   { dia <- 0;}
 }
 se( dia == 0)
 { imprima "\n Tipo de usuario desconhecido";}
 senao
 { imprima "\nnome do livro: ", livro;
   imprima "\nnome tipo de usuário: ", tipo;
   imprima "\ntotal de dias: ", dia;
 }
 imprima "\n";
fimprog
```

algoritmo 146

Fazer um algoritmo que leia o percurso em quilômetros, o tipo do carro e informe o consumo estimado de combustível, sabendo-se que um carro tipo C faz 12 km com um litro de gasolina, um tipo B faz 9 km e o tipo C, 8 km por litro.

```
prog se57
 real percurso, consumo;
 string tipo;
 imprima "\ndigite tipo de carro( A / B / C): ";
 leia  tipo;
 imprima "\ndigite percurso: ";
 leia  percurso;
 se ( tipo == "C" || tipo == "c")
 {  consumo <- percurso / 12;}
 senao
 {
  se ( tipo == "B" || tipo == "b")
  {  consumo <- percurso / 10;  }
  senao
  {
    se ( tipo == "A" || tipo == "a")
```

```
    {  consumo <- percurso / 8;  }
    senao
    {  consumo <- 0.;  }
  }
 }
 se ( consumo < > 0.)
 {  imprima "\nConsumo estimado em litros: ", formatar(consumo,2);}
 senao
 {  imprima "\nModelo inexistente";}
 imprima "\n";
fimprog
```

algoritmo 147

Criar um algoritmo que informe a quantidade total de calorias de uma refeição a partir da escolha do usuário que deverá informar o prato, a sobremesa e bebida (veja a tabela a seguir).

PRATO	*SOBREMESA*	*BEBIDA*
Vegetariano 180cal	*Abacaxi 75cal*	*Chá 20cal*
Peixe 230cal	*Sorvete diet 110cal*	*Suco de laranja 70cal*
Frango 250cal	*Mousse diet 170cal*	*Suco de melão 100cal*
Carne 350cal	*Mousse chocolate 200cal*	*Refrigerante diet 65cal*

```
prog se58
 int op, os, ob, calorias;
 calorias <- 0;
 imprima "\nDigite 1- Vegetariano  2- Peixe  3- Frango  4- Carne: ";
 leia op;
 se(op==1)
 {calorias <- calorias +180;}
 senao
 { se(op==2)
  {calorias <- calorias +230;}
  senao
  {se(op==3)
   {calorias <- calorias +250;}
   senao
   {se(op==4)
    {calorias <- calorias +350;}
   }
  }
 }
 imprima "\nDigite 1- Abacaxi  2- Sorvete diet  3- Suco melao  4-
refrigerante diet: ";
 leia op;
```

```
 se(op==1)
 {calorias <- calorias +75;}
 senao
 { se(op==2)
  {calorias <- calorias +110;}
  senao
  {se(op==3)
   {calorias <- calorias +170;}
   senao
   {se(op==4)
    {calorias <- calorias +200;}
   }
  }
 }
 imprima "\nDigite 1- Cha  2- Suco de laranja  3- Mousse diet  4- Mousse
de chocolate: ";
 leia op;
 se(op==1)
 {calorias <- calorias +20;}
 senao
 { se(op==2)
  {calorias <- calorias +70;}
  senao
  {se(op==3)
   {calorias <- calorias +100;}
   senao
   {se(op==4)
    {calorias <- calorias +65;}
   }
  }
 }
 imprima "\n Total de calorias: ", calorias;
 imprima "\n";
fimprog
```

algoritmo 148

Criar um algoritmo que leia o destino do passageiro, se a viagem inclui retorno (ida e volta) e informar o preço da passagem conforme a tabela a seguir.

DESTINO	*IDA*	*IDA E VOLTA*
Região Norte	*R$500,00*	*R$900,00*
Região Nordeste	*R$350,00*	*R$650,00*
Região Centro-Oeste	*R$350,00*	*R$600,00*
Região Sul	*R$300,00*	*R$550,00*

```
prog se59
 int op, iv;
 imprima "\nDigite 1- Regiao Norte  2- Regiao Nordeste  3- Regiao Centro-Oeste  4-
 Regiao Sul: ";
 leia op;
 imprima "\nDigite 1- Ida  2- Ida e Volta: ";
 leia iv;
 se(op == 1)
 { se(iv == 1)
  { imprima "\nO valor da passagem de ida para R.Norte R$500.00";}
  senao
  { imprima "\nO valor da passagem de ida-volta para R.Norte: R$950.00";}
 }
 senao
 { se(op == 2)
  { se(iv == 1)
    { imprima "\nO valor da passagem de ida para R.Nordeste: R$350.00";}
    senao
    { imprima "\nO valor da passagem de ida-volta para R.Nordeste: R$650.00";}
   }
   senao
   { se(op == 3)
     { se(iv == 1)
       { imprima "\nO valor da passagem de ida para R.C.Oeste: R$350.00";}
       senao
       { imprima "\nO valor da passagem de ida-volta para R.C.Oeste: R$600.00";}
      }
      senao
      { se(op == 4)
        { se(iv == 1)
          { imprima "\nO valor da passagem de ida para R.Sul: R$300.00";}
          senao
          { imprima "\nO valor da passagem de ida-volta para R.Sul: R$550.00";}
         }
         senao
         { imprima "\nOpcao Inexistente";}
      }
   }
 }
 imprima "\n";
fimprog
```

algoritmo 149

Um comerciante calcula o valor da venda, tendo em vista a tabela a seguir:

VALOR DA COMPRA	*VALOR DA VENDA*
Valor < R$ 10,00	*lucro de 70%*
R$10,00 ≤ valor < R$ 30,00	*lucro de 50%*
R$30,00 ≤ valor < R$ 50,00	*lucro de 40%*
Valor ≥ R$50,00	*lucro de 30%*

Criar o algoritmo que possa entrar com nome do produto e valor da compra e imprimir o nome do produto e o valor da venda.

```
prog se60
 real  vcompra, vvenda;
 string nomep;
 imprima "\ndigite nome do produto: ";
 leia  nomep;
 imprima "\ndigite valor da compra: ";
 leia  vcompra;
 se ( vcompra < 10. )
 {  vvenda <- vcompra * 1.7;}
 senao
 {
  se ( vcompra < 30. )
  {  vvenda <- vcompra * 1.5; }
  senao
  {
    se ( vcompra < 50.)
    {  vvenda <- vcompra * 1.4; }
    senao
    {  vvenda <- vcompra * 1.3;     }
  }
 }
 imprima "\n", nomep, "ƀsera vendido por R$ƀ", formatar(vvenda +
0.001,2);
 imprima "\n";
fimprog
```

algoritmo 150

Criar um algoritmo que leia um ângulo em graus e apresente:

- *o seno do ângulo, se o ângulo pertencer a um quadrante par;*
- *o co-seno do ângulo, se o ângulo pertencer a um quadrante ímpar.*

```
prog se61
 real  ang, rang;
 imprima "\ndigite angulo em graus: ";
 leia ang;
 rang <-  ang * pi / 180 ;
 se((rang > pi/2 && rang <= pi) || (rang > 3*pi/2 && rang <= 2*pi))
 {  imprima "\nseno: ", sen(rang); }
 senao
 {  imprima "\nco-seno: ", cos(rang);}
 imprima "\n";
fimprog
```

algoritmo 151

Um endocrinologista deseja controlar a saúde de seus pacientes e, para isso, se utiliza do Índice de Massa Corporal (IMC). Sabendo-se que o IMC é calculado através da seguinte fórmula:

$$IMC = \frac{peso}{altura^2}$$

Onde:

– peso é dado em Kg

– altura é dada em metros

Criar um algoritmo que apresente o nome do paciente e sua faixa de risco, baseando-se na seguinte tabela:

IMC	*FAIXA DE RISCO*
abaixo de 20	*abaixo do peso*
a partir de 20 até 25	*normal*
acima de 25 até 30	*excesso de peso*
acima de 30 até 35	*obesidade*
acime de 35	*obesidade mórbida*

```
prog se62
 real  peso, altura, imc;
 imprima "\ndigite peso: ";
 leia peso;
 imprima "\ndigite altura: ";
 leia altura;
 imc <-  peso / altura **2;
 se ( imc < 20.)
 {   imprima "\nabaixo do peso";}
 senao
 {
  se ( imc <=25. )
  {  imprima "\nnormal";  }
  senao
  {
    se ( imc <=30. )
    {  imprima "\nexcesso de peso "; }
    senao
    {
       se ( imc <=35. )
       {  imprima "\nobesidade ";  }
       senao
       {   imprima "\no obesidade mórbida";   }
    }
  }
 }
 imprima "\n";
fimprog
```

algoritmo 152

Criar um algoritmo que a partir da idade e peso do paciente calcule a dosagem de determinado medicamento e imprima a receita informando quantas gotas do medicamento o paciente deve tomar por dose. Considere que o medicamento em questão possui 500 mg por ml, e que cada ml corresponde a 20 gotas.

- *Adultos ou adolescentes desde 12 anos, inclusive, se tiverem peso igual ou acima de 60 quilos devem tomar 1000 mg; com peso abaixo de 60 quilos devem tomar 875 mg.*
- *Para crianças e adolescentes abaixo de 12 anos é dosagem é calculada pelo peso corpóreo conforme a tabela a seguir:*

5 kg a 9 kg = 125 mg	*24.1 kg a 30 kg = 500 mg*
9.1 kg a 16 kg = 250 mg	*acima de 30 kg = 750 mg*
16.1 kg a 24 kg = 375 mg	

```
prog se63
 int idade;
 real  peso,gotas;
 imprima "\ndigite peso: ";
 leia peso;
 imprima "\ndigite idade: ";
 leia idade;
 gotas <- 500  /  20;  # calculo do número de mg por gotas
 se (peso <5.)
 {  imprima "\nnao pode tomar o remedio porque nao tem peso. Consulte
medico."; }
 senao
 {
   se ( idade >= 12)
   {
     se ( peso >= 60. )
     { imprima "\nTomar  ",1000 / gotas, "  gotas"; }
     senao
     {  imprima "\nTomar  ", 875  /gotas, "  gotas"; }
   }
   senao
   {
    se ( peso <= 9.)
    {  imprima "\nTomar  ", 125  / gotas, "  gotas";  }
    senao
    {
     se ( peso <= 16. )
     {   imprima "\nTomar  ", 250  / gotas, "  gotas";  }
     senao
     {
```

```
      se ( peso <= 24. )
      { imprima "\nTomar ", 375 / gotas, " gotas"; }
      senao
      {
        se ( peso <=30. )
        { imprima "\nTomar ", 500 / gotas, " gotas"; }
        senao
        { imprima "\nTomar ", 750 / gotas, " gotas"; }
       }
     }
   }
 }
}
imprima "\n";
fimprog
```

algoritmo 153

O prefeito do Rio de Janeiro contratou uma firma especializada para manter os níveis de poluição considerados ideiais para um país do 1º mundo. As indústrias, maiores responsáveis pela poluição, foram classificadas em três grupos. Sabendo-se que a escala utilizada varia de 0,05 e que o índice de poluição aceitável é até 0,25, fazer um algoritmo que possa imprimir intimações de acordo com o índice e a tabela a seguir:

índice	*indústrias que receberão intimação*
0,3	*1º grupo*
0,4	*1º e 2º grupos*
0.5	*1º, 2º e 3º grupos*

```
prog se64
 real indice;
 imprima "\ndigite indice de poluicao: ";
 leia indice;
 se ( indice >= 0.5 )
 { imprima "\nsuspender atividades as industrias dos grupos 1, 2 e 3";}
 senao
 {
  se ( indice >= 0.4 )
  { imprima "\nsuspender atividades as industrias dos grupos 1 e 2 "; }
  senao
  {
    se ( indice >= 0.3 )
    { imprima "\nsuspender atividades as industrias dos grupos 1 "; }
    senao
    { imprima "\nindice de poluicao aceitavel para todos os grupos "; }
  }
 }
 imprima "\n";
fimprog
```

algoritmo 154

A polícia rodoviária resolveu fazer cumprir a lei e cobrar dos motoristas o DUT. Sabendo-se que o mês em que o emplacamento do carro deve ser renovado é determinado pelo último número da placa do mesmo, criar um algoritmo que, a partir da leitura da placa do carro, informe o mês em que o emplacamento deve ser renovado.

↳ *Leia a placa do carro em uma variável caracter.*

```
prog se65
 string placa, p;
 imprima "\ndigite placa: ";
 leia  placa ;
 p <- strult(placa);
 se ( p  == "1" )
 { imprima "\njaneiro"; }
 senao
 { se ( p  == "2" )
   {  imprima "\nfevereiro"; }
   senao
   { se ( p  == "3" )
     { imprima "\nmarco"; }
     senao
     { se ( p  == "4" )
        { imprima "\nabril"; }
        senao
        { se ( p  == "5" )
          { imprima "\nmaio"; }
          senao
          { se ( p  == "6" )
            { imprima "\njunho"; }
            senao
            { se ( p  == "7" )
              { imprima "\njulho";  }
              senao
              { se ( p  == "8" )
                {  imprima "\nagosto"; }
                senao
                {  se ( p  == "9" )
                  {  imprima "\nsetembro";  }
                  senao
                  {  se ( p  == "0" )
                    { imprima "\noutubro"; }
                    senao
                    { imprima "\nopcao invalida"; }
```

```
                  }
                }
              }
            }
          }
        }
      }
   }
 }
 imprima "\n";
fimprog
```

algoritmo 155

Ler uma palavra e, se ela começar pela letra L ou D (também deve ser considerado l ou d), formar uma nova palavra que terá os dois primeiros caracteres e o último, caso contrário a nova palavra será formada por todos os caracteres menos o primeiro.

```
prog se66
 string pal, pal1, pal2, letra;
 int tam;
 imprima "\ndigite palavra: ";
 leia pal;
 letra <- strprim(pal);
 se(  letra =="l" || letra =="L"  || letra  =="d"  || letra  =="D"  )
 { pal1 <-strnprim(pal,2);
   pal2 <-strult(pal);
   pal1 <-strconcat(pal1,pal2 );}
 senao
 {pal1 <-strresto(pal);}
 imprima "\npalavra: ", pal1;
 imprima "\n";
fimprog
```

algoritmo 156

Criar um algoritmo que leia uma data (dia, mês e ano em separado) e imprima se a data é válida ou não.

```
prog se67
 int ANO, MES, DIA, VD;
 imprima "\ndigite dia: ";
 leia DIA;
 imprima "\ndigite mes: ";
 leia MES;
 imprima "\ndigite ano: ";
 leia ANO;
 se(ANO>=1)
```

```
{ VD <- 1;
  se( MES < 1 || MES > 12 || DIA < 1 || DIA > 31)
  { VD <- 0; }
  senao
  {
    se( (MES == 4 || MES == 6 || MES == 9 || MES == 11) && DIA > 30)
    {VD <- 0;}
    senao
    { se(MES == 2)
      {
        se((ANO % 4==0 && ANO % 100 < > 0) || ANO % 400==0)
        {
          se(DIA > 29)
          { VD <- 0;}
        }
        senao
        {
          se( DIA > 28)
          { VD <- 0;}
        }
      }
    }
  }
}
senao
{ VD <- 0;}
se( VD == 0 )
{ imprima "\nDATA INVALIDA";}
senao
{ imprima "\nDATA VALIDA";}
imprima "\n";
fimprog
```

algoritmo 157

Criar um algoritmo que leia uma data (no formato ddmmaaaa) e imprimir se a data é válida ou não.

```
prog se68
 int DATA, ANO, MES, DIA, VD;
 imprima "\ndata no formato ddmmaaaa: ";
 leia DATA;
 DIA <- DATA div 1000000;
 MES <- DATA % 1000000 div 10000;
 ANO <- DATA % 10000;
 se(ANO >=1)
 {  VD <- 1;
    se(MES > 12 || DIA < 1 || DIA > 31)
    { VD <- 0; }
```

```
  senao
  {
    se((MES == 4||MES == 6||MES == 9||MES == 11)&& DIA > 30)
    {VD <- 0;}
    senao
    {
      se(MES == 2)
      {
        se((ANO % 4==0 && ANO % 100 < > 0) || ANO % 400==0)
        {
          se(DIA > 29)
          { VD <- 0;}
         }
         senao
         { se( DIA > 28)
           { VD <- 0;}
         }
      }
   }
 }
}
senao
{ VD <- 0;}
se( VD == 0 )
{imprima "\nDATA INVALIDA";}
senao
{imprima "\nDATA VALIDA";}
imprima "\n";
fimprog
```

algoritmo 158

Criar um algoritmo que entre com o valor de x, calcule e imprima o valor de f(x).

$$f(x) = \frac{8}{2 - x}$$

```
prog se69
 real x, fx;
 imprima "\ndigite valor de x: ";
 leia x;
 se(x < > 2. )
 {fx <- 8  / (2 - x);
 imprima "\nf(x)= ",fx;
 }
 senao
 {imprima "\nNAO PODE SER FEITA";}
 imprima "\n";
fimprog
```

algoritmo 159

Criar um algoritmo que entre com o valor de x, calcule e imprima o valor de f(x).

$$f(x) = \frac{5x+3}{\sqrt{x^2 - 16}}$$

```
prog se70
 real x, fx;
 imprima "\ndigite valor de x: ";
 leia x;
 se(x > 4. || x < (-4.))
 {fx <- (5 * x + 3 ) / raiz(x**2 -16);
 imprima "\nf(x)= ",fx;
 }
 senao
 {imprima "\nNAO PODE SER FEITA";}
 imprima "\n";
fimprog
```

algoritmo 160

Entrar com o valor de x e imprimir y:

$$Y = f(x) \begin{cases} 1, se\ x <= 1 \\ 2, se\ 1 < x <= 2 \\ x^2, se\ 2 < x <= 3 \\ x^3, se\ x > 3 \end{cases}$$

```
prog se71
 real x, y;
 imprima "\ndigite valor de x: ";
 leia x;
 se(x <= 1.)
 {y <- 1.;}
 senao
 { se(x <= 2.)
  {y <- 2.;}
  senao
  { se(x <= 3.)
    {y <- x **2;}
    senao
    {y <- x **3;}
  }
 }
 imprima "\nvalor de f(x): ",y;
 imprima "\n";
fimprog
```

Capítulo 4

Estruturas de repetição: para, enquanto e faca-enquanto

Este capítulo é um dos mais importantes do livro. A partir desse ponto, os alunos começam a ter muita dificuldade na resolução dos algoritmos por não entenderem onde as estruturas de repetição deverão ser colocadas para atender às exigências dos enunciados. Uma das perguntas mais comuns é: "o comendo **leia** ven antes ou depois da estrutura de repetição?" Leia com muita atenção!

Nossos algoritmos precisarão ser executados mais de uma vez e, para que não tenhamos de reescrever trechos idênticos que aumentariam consideravelmente o tamanho, utilizamos três estruturas de repetição.

ESTRUTURA DO PARA

Usada quando o número de repetições for conhecido durante a elaboração do algoritmo ou quando puder ser fornecido durante a execução.

Ex: Criar um algoritmo que possa entrar com a nota de 20 alunos e imprimir a média da turma.

```
para  ( valor inicial ;  <condição >; <valor do incremento> )
{
      bloco de comandos
}
```

< valor inicial>; – é uma expressão do tipo:

```
<identificador> <- <valor inicial> ;
```

Exemplos:

a <- 0; A variável **a** recebe o valor inicial **0**.

c <- 100; A variável **c** recebe o valor inicial **100**.

x <- strtam(pal) -1; A variável **x** recebe o valor inicial que é igual ao número de caracteres da variável **pal** menos 1.

< condição >; – é uma expressão do tipo:

```
<identificador> <=, <, > ou >= <valor final> ;
```

Exemplos:

a <= 10; A variável **a** poderá receber valores enquanto forem menores ou iguais a **10**.

c >= 2; A variável **c** poderá receber valores enquanto forem maiores ou iguais a **2**.

x >= 0; A variável **x** poderá receber valores enquanto forem maiores ou iguais a **0**.

< valor do incremento> - é uma expressão do tipo:

```
<identificador> <- <identificador> + ou - valor
```

Exemplos:

a <- a + 1; A variável **a** é incrementada de **1**. /* contador */

c <- c - 2; A variável **c** é decrementada de **2**.

x <- x - 1; A variável **x** é decrementada de **1**.

Já vimos que os operadores ++ e -- são equivalentes aos comandos de atribuição quando a variável é incrementada ou decrementada de **1**; por isso, o primeiro e o terceiro exemplos poderiam ser escritos assim:

a ++; A variável **a** é incrementada em **1**.

x --; A variável **x** é decrementada de **1**.

Observações

1. O identificador tem de ser do tipo **int**.

2. Os valores inicial e final poderão ser constantes numéricas inteiras, funções que retornem números inteiros ou expressões que retornem números inteiros.
3. O valor que é incrementado ou decrementado da variável poderá ser constante ou uma outra variável.
4. Esta estrutura é a própria expressão de uma PA, pois ela gera valores automaticamente na MP, veja bem:

para (a <- 1 ; a <= 5; a ++)	**PA** 1 : 2 : 3 : 4 : 5
MP c ~~1~~ ~~2~~ ~~3~~ 4 ~~5~~ 6	sabendo-se que a fórmula para calcular o enésimo termo de uma PA é: $a_n = a_1 + (n - 1).r$, temos: **$a_1 = 1$ $a_n = 5$**

5. O número de repetições do bloco de comandos será igual **ao NÚMERO DE TERMOS DA SÉRIE**; portanto, na maioria das vezes, não importam os valores que a variável que controla a repetição assumirá.
6. A variável que controla a repetição deverá ser declarada e poderá ser impressa se precisarmos dela para numerar uma lista, posicionar etc.
7. A variável que controla a repetição **JAMAIS** aparecerá num comando de **leitura** dentro do bloco de repetição.

Vejamos alguns exemplos:

algoritmo 161

Criar um algoritmo que entre com cinco números e imprimir o quadrado de cada número.

```
prog exemplo
 int  c;
 real num;
 para( c<- 1; c <= 5 ; c++)
 {
    imprima "\nnumero: ";
    leia  num ;
    imprima "quadrado: ", num ** 2;
 }
 imprima "\n";
fimprog
```

MP c	MP num	Saída
~~1~~	~~2~~	numero:2.
~~2~~	~~4~~	4.0
~~3~~	~~5~~	numero:4.
~~4~~	~~8~~	16.0
~~5~~	12	numero:5.
6		25.0
		numero:8.
		64.0
		numero:12.
		144.0

Vamos explicar:

1. Após serem reservadas posições na MP para as variáveis **c** e **num**, é executado o comando **para.**
2. O primeiro argumento c<- 1 só é executado uma vez antes do loop.
3. Os comandos **imprima, leia** e **imprima** são executados pela primeira vez.
4. Após }, o comando c++ é executado.
5. O comando c <= 5 é executado e enquanto a variável **c** estiver armazenando um valor menor ou igual a 5, o bloco se repete; mas, quando o valor ultrapassar 5, como no exemplo 6, o fluxo do algoritmo passa para o primeiro comando após }.

algoritmo 162

Criar um algoritmo que imprima todos os números pares no intervalo 1-10.

```
prog exemplo
 int  c;
 para( c<- 2; c <= 10 ; c<- c + 2)
 { imprima "\n", c; }
 imprima "\n";
fimprog
```

MP c	Saída
~~2~~	2
~~4~~	4
~~6~~	6
~~8~~	8
~~10~~	10
12	

1. Após ser reservada uma posição na MP para a variável **c**, o comando **para** é executado.

2. O primeiro argumento c<- 2 só é executado uma vez antes do loop.

3. O comando **imprima "\n"**,c; é executado pela primeira vez.

4. Após }, o comando c<- c + 2 é executado.

5. O comando c <= 10 é executado e enquanto a variável **c** estiver armazenando um valor menor ou igual a 10, o bloco se repete; mas, quando o valor ultrapassar 10, como no exemplo **12**, {o fluxo do algoritmo passa para o primeiro comando após }.

6. *Nenhuma série tem entrada de dados no bloco de repetição.*

PARA dentro de PARA

Imagine você dando uma volta na Lagoa Rodrigo de Freitas (Rio de Janeiro): *passa uma vez pelo Corte do Cantagalo, uma vez pelo Clube do Flamengo, uma vez pelo Clube Piraquê, uma vez pelos fundos do Clube Militar*; esse trajeto representa seus algoritmos feitos até agora com os comandos aprendidos.

Imagine que seu preparo físico melhorou e agora você consegue dar três ou cinco voltas ou, quem sabe, até dez voltas na Lagoa; isso significa que você passará dez vezes pelos mesmos lugares. Esse exemplo representa a estrutura do Para.

Imagine agora que você, orientado por um profissional especializado, passou a dar três voltas na Lagoa mas, em frente ao Clube do Flamengo, você fará cinco abdominais. Se, em cada volta você faz cinco abdominais, ao final, você terá feito 15 abdominais e terá dado três voltas na Lagoa. Isso representa um Para dentro de um PARA. Acompanhe:

voltaLagoa	abdominais	
1		1ª volta na Lagoa
	~~1~~	1º abdominal
	~~2~~	2º abdominal
	~~3~~	3º abdominal
	4	4º abdominal
	5	5º abdominal
	6	**Quando ia contar o 6º abdominal, PAROU e voltou a correr**

voltaLagoa	abdominais	
2	 ~~1~~ ~~2~~ ~~3~~ 4 ~~5~~ ~~6~~	2ª volta na Lagoa 1º abdominal 2º abdominal 3º abdominal 4º abdominal 5º abdominal **Quando ia contar o 6º abdominal, PAROU e voltou a correr**
3 ~~3~~ 4	 ~~1~~ ~~2~~ ~~3~~ 4 ~~5~~ 6 ~~6~~	3ª volta na Lagoa 1º abdominal 2º abdominal 3º abdominal 4º abdominal 5º abdominal **Quando ia contar o 6º abdominal, PAROU e voltou a correr** **Quando ia contar a 4ª volta, PAROU e foi para casa**

Vejamos como poderíamos representar em algoritmo:

algoritmo 163

```
prog corredor
 int voltaLagoa, abdominais;
 para( voltaLagoa <- 1 ; voltaLagoa <=3 ; voltaLagoa++)
 {  imprima "\n",voltaLagoa,"a volta na Lagoa";
   para( abdominais<- 1 ; abdominais <=5; abdominais++)
   { imprima "\n  ", abdominais, "o abdominal";    }
 }
 imprima "\n";
fimprog
```

Você pode observar que, cada vez que chegava em frente ao Clube do Flamengo, começava a contar a partir do 1 seus abdominais e só parava quando sua próxima contagem fosse 6.

DESAFIO

- *Você já pensou em simular uma pausa usando a estrutura do* ***para****?*

```
Comecei
<após alguns segundos>
Parei
```

algoritmo 164

```
prog desafio1
 int a;
 imprima "\nComecei\n";
 para(a<-1; a<=50000;a++)
 { }
 imprima "\nFim";
 imprima "\n";
fimprog
```

■ *Você já pensou em fazer várias tabulações usando a estrutura do **para** ?*

```
1                                      63
Zona1
                                       Zona9
```

algoritmo 165

```
prog desafio2
 int a;
 imprima "zona1\n";
 para(a<-1; a<=8;a++)
 { imprima "\t";}
 imprima "zona9";
 imprima "\n";
fimprog
```

■ *Você já pensou em triangular uma matriz de ordem **10** usando a estrutura do **para**?*

TODOS

1-1	*1-2*	*1-3*	*1-4*	*1-5*	*1-6*	*1-7*	*1-8*	*1-9*	*1-10*
2-1	*2-2*	*2-3*	*2-4*	*2-5*	*2-6*	*2-7*	*2-8*	*2-9*	*2-10*
3-1	*3-2*	*3-3*	*3-4*	*3-5*	*3-6*	*3-7*	*3-8*	*3-9*	*3-10*
4-1	*4-2*	*4-3*	*4-4*	*4-5*	*4-6*	*4-7*	*4-8*	*4-9*	*4-10*
5-1	*5-2*	*5-3*	*5-4*	*5-5*	*5-6*	*5-7*	*5-8*	*5-9*	*5-10*
6-1	*6-2*	*6-3*	*6-4*	*6-5*	*6-6*	*6-7*	*6-8*	*6-9*	*6-10*
7-1	*7-2*	*7-3*	*7-4*	*7-5*	*7-6*	*7-7*	*7-8*	*7-9*	*7-10*
8-1	*8-2*	*8-3*	*8-4*	*8-5*	*8-6*	*8-7*	*8-8*	*8-9*	*8-10*
9-1	*9-2*	*9-3*	*9-4*	*9-5*	*9-6*	*9-7*	*9-8*	*9-9*	*9-10*
10-1	*10-2*	*10-3*	*10-4*	*10-5*	*10-6*	*10-7*	*10-8*	*10-9*	*10-10*

algoritmo 166

```
prog desafio31
 int L,c,t;
 imprima "\nTODOS\n";
 para(L<-1;L<=10;L++)
 {
  para(c<-1;c<=10;c++)
   { imprima L, "-",c, "\t" ; }
   imprima "\n";
 }
 imprima "\n";
fimprog
```

ACIMA DA DIAGONAL PRINCIPAL

1-2	*1-3*	*1-4*	*1-5*	*1-6*	*1-7*	*1-8*	*1-9*	*1-10*
	2-3	*2-4*	*2-5*	*2-6*	*2-7*	*2-8*	*2-9*	*2-10*
		3-4	*3-5*	*3-6*	*3-7*	*3-8*	*3-9*	*3-10*
			4-5	*4-6*	*4-7*	*4-8*	*4-9*	*4-10*
				5-6	*5-7*	*5-8*	*5-9*	*5-10*
					6-7	*6-8*	*6-9*	*6-10*
						7-8	*7-9*	*7-10*
							8-9	*8-10*
								9-10

algoritmo 167

```
prog desafio32
 int L,c,t;
 imprima "\nACIMA DA DIAGONAL PRINCIPAL\n";
 para(L<-1;L<=9;L++)
 {
  para(c<-L+1;c<=10;c++)
   {  imprima "\t",L, "-",c; }
   imprima "\n";
   para(t<-1;t<=L;t++)
   {imprima "\t";}
 }
 imprima "\n";
fimprog
```

DIAGONAL PRINCIPAL

1-1

2-2

3-3

4-4

5-5

6-6

7-7

8-8

9-9

10-10

algoritmo 168

```
prog desafio33
 int L,c,t;
 imprima "\nDIAGONAL PRINCIPAL\n";
 para(L<-1;L<=10;L++)
 {
  imprima L, "-",L,"\n";
  para(t<-1;t<=L;t++)
   {  imprima "\t";  }
 }
 imprima "\n";
fimprog
```

ABAIXO DA DIAGONAL PRINCIPAL

2-1

3-1 3-2

4-1 4-2 4-3

5-1 5-2 5-3 5-4

6-1 6-2 6-3 6-4 6-5

7-1 7-2 7-3 7-4 7-5 7-6

8-1 8-2 8-3 8-4 8-5 8-6 8-7

9-1 9-2 9-3 9-4 9-5 9-6 9-7 9-8

10-1 10-2 10-3 10-4 10-5 10-6 10-7 10-8 10-9

algoritmo 169

```
prog desafio34
 int L,c,t;
 imprima "\nABAIXO DA DIAGONAL PRINCIPAL\n";
 para(L<-2;L<=10;L++)
 {
  imprima "\n";
  para(c<-1;c < L;c++)
   {  imprima L, "-",c,"\t";  }
 }
 imprima "\n";
fimprog
```

ACIMA DA DIAGONAL SECUNDÁRIA

1-1	*1-2*	*1-3*	*1-4*	*1-5*	*1-6*	*1-7*	*1-8*	*1-9*
2-1	*2-2*	*2-3*	*2-4*	*2-5*	*2-6*	*2-7*	*2-8*	
3-1	*3-2*	*3-3*	*3-4*	*3-5*	*3-6*	*3-7*		
4-1	*4-2*	*4-3*	*4-4*	*4-5*	*4-6*			
5-1	*5-2*	*5-3*	*5-4*	*5-5*				
6-1	*6-2*	*6-3*	*6-4*					
7-1	*7-2*	*7-3*						
8-1	*8-2*							
9-1								

algoritmo 170

```
prog desafio35
 int L,c,t;
 imprima "\nACIMA DA DIAGONAL SECUNDARIA\n";
 para(L<-1;L<=9;L++)
 {
  para(c<-1;c<11 - L;c++)
  {  imprima L, "-",c,"\t";  }
  imprima "\n";
 }
 imprima "\n";
fimprog
```

DIAGONAL SECUNDÁRIA

```
                                                                        1-10
                                                                2-9
                                                        3-8
                                                4-7
                                        5-6
                                6-5
                        7-4
                8-3
        9-2
10-1
```

algoritmo 171

```
prog desafio36
 int L,c,t;
 imprima "\nDIAGONAL SECUNDARIA\n";
 para(L<-1;L<=10;L++)
 {
  para(t<-10 - L;t >=1;t--)
  {  imprima "\t";  }
   imprima L, "-",11-L,"\n";
 }
 imprima "\n";
fimprog
```

ABAIXO DA DIAGONAL SECUNDARIA

```
                                                                        2-10
                                                                3-9     3-10
                                                        4-8     4-9     4-10
                                                5-7     5-8     5-9     5-10
                                        6-6     6-7     6-8     6-9     6-10
                                7-5     7-6     7-7     7-8     7-9     7-10
                        8-4     8-5     8-6     8-7     8-8     8-9     8-10
                9-3     9-4     9-5     9-6     9-7     9-8     9-9     9-10
        10-2    10-3    10-4    10-5    10-6    10-7    10-8    10-9    10-10
```

algoritmo 172

```
prog desafio37
 int L,c,t;
 imprima "\nABAIXO DA DIAGONAL SECUNDÁRIA\n";
 imprima "\n";
 para(L<-2;L<=10;L++)
 {
   para(t<-11-L;t >=1;t--)
   {imprima "\t"; }
  para(c<-12-L;c<=10;c++)
  {  imprima L, "-",c,"\t";  }
  imprima "\n";
 }
 imprima "\n";
fimprog
```

E se a matriz fosse assim?

0-0	*0-1*	*0-2*	*0-3*	*0-4*	*0-5*	*0-6*	*0-7*	*0-8*	*0-9*
1-0	*1-1*	*1-2*	*1-3*	*1-4*	*1-5*	*1-6*	*1-7*	*1-8*	*1-9*
2-0	*2-1*	*2-2*	*2-3*	*2-4*	*2-5*	*2-6*	*2-7*	*2-8*	*2-9*
3-0	*3-1*	*3-2*	*3-3*	*3-4*	*3-5*	*3-6*	*3-7*	*3-8*	*3-9*
4-0	*4-1*	*4-2*	*4-3*	*4-4*	*4-5*	*4-6*	*4-7*	*4-8*	*4-9*
5-0	*5-1*	*5-2*	*5-3*	*5-4*	*5-5*	*5-6*	*5-7*	*5-8*	*5-9*
6-0	*6-1*	*6-2*	*6-3*	*6-4*	*6-5*	*6-6*	*6-7*	*6-8*	*6-9*
7-0	*7-1*	*7-2*	*7-3*	*7-4*	*7-5*	*7-6*	*7-7*	*7-8*	*7-9*
8-0	*8-1*	*8-2*	*8-3*	*8-4*	*8-5*	*8-6*	*8-7*	*8-8*	*8-9*
9-0	*9-1*	*9-2*	*9-3*	*9-4*	*9-5*	*9-6*	*9-7*	*9-8*	*9-9*

algoritmo 173

```
prog  desafio4
 int L,c,t;
 imprima "\nTODOS\n";
 para(L<-0;L<=9;L++)
 {
  para(c<-0;c<=9;c++)
   { imprima L, "-",c, "\t" ;  }
   imprima "\n";
 }
 imprima "\n";
fimprog
```

EXERCÍCIOS – LISTA 3

ESTRUTURA DO PARA

algoritmo 174

Imprimir todos os números de 1 até 100.

```
prog para1
 int i;
 para( i <- 1; i <= 100; i++ )
 {   imprima i, "   "; }
 imprima "\n";
fimprog
```

algoritmo 175

Imprimir todos os números de 100 até 1.

```
prog para2
 int i;
 para( i <- 100; i >= 1; i-- )
 {   imprima i, "   "; }
 imprima "\n";
fimprog
```

algoritmo 176

Imprimir os 100 primeiros pares.

```
prog para3
 int i;
 para( i <- 2; i <= 200; i <- i+2 )
 {   imprima i, "   "; }
 imprima "\n";
fimprog
```

algoritmo 177

Imprimir os múltiplos de 5, no intervalo de 1 até 500.

```
prog para4
 int i;
 para( i <- 5; i <= 500; i <- i +5 )
 { imprima i, "   "; }
 imprima "\n";
fimprog
```

algoritmo 178

Imprimir o quadrado dos números de 1 até 20.

```
prog para5
 int c;
 para( c <- 1; c <= 20; c++)
 { imprima c ^ 2, "   " ; }
 imprima "\n";
fimprog
```

algoritmo 179

Criar um algoritmo que imprima os números pares no intervalo de 1 a 600.

```
prog para6
 int i;
 para( i <- 2; i <= 600; i <- i+2 )
 {   imprima i, "   "; }
 imprima "\n";
fimprog
```

algoritmo 180

Criar um algoritmo que imprima os números de 120 a 300.

```
prog para7
 int c;
 para( c <- 120; c <=300; c++)
 {   imprima c , "   " ; }
 imprima "\n";
fimprog
```

algoritmo 181

Criar um algoritmo que imprima todos os números de 1 até 100 e a soma deles.

```
prog para8
 int i, soma;
 soma <-0;
 para( i <-1; i <= 100;  i ++)
 {
   soma <- soma + i;
   imprima i, "  ";
 }
 imprima "\nSomatorio de 1 a 100: ",soma;
 imprima "\n";
fimprog
```

algoritmo 182

Entrar com 10 números e imprimir a metade de cada número.

```
prog para9
 int a;
 real numero;
 para( a <- 1; a <=10; a++)
 { imprima "\ndigite numero: ";
   leia numero;
   imprima  "metade: ", numero / 2;
 }
 imprima "\n";
fimprog
```

algoritmo 183

Entrar com 10 números e imprimir o quadrado de cada número.

```
prog para10
 int a;
 real numero;
 para( a <- 1; a <=10; a++)
 {
  imprima "\ndigite numero: ";
  leia numero;
  imprima  "quadrado: ", numero ** 2;
 }
 imprima "\n";
fimprog
```

algoritmo 184

Entrar com 8 números e, para cada número, imprimir o logaritmo desse número na base 10.

```
prog para11
 int a;
 real numero;
 para( a <- 1; a <=8; a++)
 {
   imprima "\ndigite numero: ";
   leia numero;
   se(numero > 0.)
   {  imprima "\nlogaritmo: ", log(numero) /log(10.); }
   senao
   { imprima "\nNao fao logaritmo de numero negativo"; }
 }
 imprima "\n";
fimprog
```

algoritmo 185

Entrar com 15 números e imprimir a raiz quadrada de cada número.

```
prog para12
 int a;
 real numero;
 para( a <- 1; a <=15; a++)
 {
    imprima "\ndigite numero: ";
    leia numero;
    se(numero >= 0.)
    { imprima  "\nraiz: ", raiz(numero); }
    senao
   {  imprima "\nNao faco raiz de numero negativo";  }
 }
 imprima "\n";
fimprog
```

algoritmo 186

Entrar com quatro números e imprimir o cubo e a raiz cúbica de cada número.

```
prog para13
 int a;
 real numero;
 para( a <- 1; a <=4; a++)
 {
   imprima "\ndigite numero: ";
   leia numero;
   imprima  "\ncubo: ", numero ** 3;
   se(numero < 0.)
   { imprima "\nNao faco raiz de numero negativo: "; }
   senao
   { imprima "\nraiz: ", numero **(1/3); }
 }
 imprima "\n";
fimprog
```

algoritmo 187

Criar um algoritmo que calcule e imprima o valor de b^n. O valor de n deverá ser maior do que 1 e inteiro e o valor de b maior ou igual a 2 e inteiro.

```
prog para14
 int base, expo, pot, i;
 imprima "\nDigite a  base inteira e maior do que 1: ";
 leia base;
 imprima "\nDigite expoente inteiro maior que 1: ";
```

```
 leia expo;
 se( base  >= 2 && expo > 1)
 {
   pot <- 1;
   para( i <- 1; i <= expo; i++)
   {
       pot <- pot * base;
   }
   imprima "\npotencia: ", pot;
 }
 senao
 { imprima "\nNao satisfazem";}
 imprima "\n";
fimprog
```

algoritmo 188

Criar um algoritmo que imprima uma tabela de conversão de polegadas para centímetros. Deseja-se que na tabela conste valores desde 1 polegada até 20 polegadas inteiras.

```
prog para15
 int L,c, ca;
 imprima "\nConversao de polegadas para centimetros \n";
 para(L<-1; L<=20; L++)
 {  imprima "\n", L, "'' equivale(m) a ",L*2.54, " cm";  }
 imprima "\n";
fimprog
```

algoritmo 189

Criar um algoritmo que imprima a tabela de conversão de graus Celsius-Fahrenheit para o intervalo desejado pelo usuário. O algoritmo deve solicitar ao usuário o limite superior, o limite inferior do intervalo e o decremento.

Fórmula de conversão: C =5 (F - 32) / 9

Exemplo:

valores lidos: *68 50 14*

impressão:	*Fahrenheit*	*Celsius*
	68	*20*
	50	*10*
	14	*-10*

```
prog para16
 int f1, f2, dec,t;
 real c;
 imprima "\nentre com a temperatura maior em Fahrenheit: ";
 leia f1;
```

```
 imprima "\nentre com a temperatura menor em Fahrenheit: ";
 leia f2 ;
 imprima "\nentre com decremento: ";
 leia dec ;
 para( t <- f1; t >= f2; t <- t - dec)
 {
  c <- 5 * (t - 32)/9;
  imprima "\ntemperatura em graus Celsius: ", c;
 }
 imprima "\n";
fimprog
```

algoritmo 190

Entrar com um nome, idade e sexo de 20 pessoas. Imprimir o nome se a pessoa for do sexo masculino e tiver mais de 21 anos.

```
prog para17
 int i, idade;
 string nome,sexo;
 para( i <- 1; i <= 10; i++ )
 {
  imprima "\ndigite nome: ";
  leia nome;
  imprima "\ndigite sexo: ";
  leia sexo;
  imprima "\ndigite idade: ";
  leia idade;
  se( (strprim(sexo) == "f" || strprim(sexo)=="F") && idade >= 21)
  {   imprima "\n", nome;   }
 }
 imprima "\n";
fimprog
```

algoritmo 191

Criar um algoritmo que leia um número que será o limite superior de um intervalo e o incremento (incr). Imprimir todos os números naturais no intervalo de 0 até esse número. Suponha que os dois números lidos são maiores do que zero. Exemplo:

Limite superior: 20 *Saída: 0 5 10 15 20*
Incremento: 5

```
prog para18
 int i, limite, incr;
 imprima "\nDigite o numero limite e o incremento pressionando enter apos
 digitar cada um: ";
 leia limite;
 leia incr;
```

```
para( i <- 0; i <= limite;  i <- i + incr)
{imprima i, "    ";}
imprima "\n";
fimprog
```

algoritmo 192

Criar um algoritmo que leia um número que será o limite superior de um intervalo e imprimir todos os números ímpares menores do que esse número. Exemplo:

Limite superior: 15 *Saída: 1 3 5 7 9 11 13*

```
prog para19
 int num, i, vf;
 imprima "\nDigite um numero: ";
 leia num;
 vf <- num  - 1;
 para( i <- 1; i <= vf;  i <- i + 2)
 { imprima i, "    "; }
 imprima "\n";
fimprog
```

algoritmo 193

Criar um algoritmo que leia um número que servirá para controlar os números pares que serão impressos a partir de 2. Exemplo:

Quantos: 4 *Saída: 2 4 6 8*

```
prog para20
 int i, num, vf;
 imprima "\nDigite um numero: ";
 leia num;
 vf <- num  * 2;
 para( i <- 2; i <= vf;  i <- i + 2)
 { imprima i, "    ";}
 imprima "\n";
fimprog
```

algoritmo 194

Criar um algoritmo que leia um número e imprima todos os números de 1 até o número lido e o seu produto. Exemplo:

número: 3 *Saída: 1 2 3*
6

```
prog para21
 int i, num, produto;
 produto <- 1;
 imprima "\nDigite um numero: ";
```

```
 leia num;
 para( i <- 1; i <= num ;  i ++)
 {
    produto<- produto * i;
    imprima i, "   ";
 }
 imprima "\nProduto de 1 a ",num,": ",produto;
 imprima "\n";
fimprog
```

algoritmo 195

Criar um algoritmo que imprima a soma dos números pares entre 25 e 200.

```
prog para22
 int soma, i;
 soma <-0;
 para( i <- 26; i <= 198;  i <- i + 2)
 {   soma <-soma + i;}
 imprima "\nSoma 26-198: ",soma;
 imprima "\n";
fimprog
```

algoritmo 196

Criar um algoritmo que leia um número (num) e imprima a soma dos números múltiplos de 5 no intervalo aberto entre 1 e num. Suponha que num será maior que zero.

Limite superior: 15 *Saída: 15*
(5 10) – múltiplos de 5

```
prog para23
 int i, num, soma;
 soma <- 0;
 imprima "\nDigite um numero maior que zero: ";
 leia num;
 num<-num-1;
 para( i <- 5; i < num;  i <- i + 5)
 {   soma <- soma + i; }
 imprima "\nSoma dos multiplos de 5: " , soma;
 imprima "\n";
fimprog
```

algoritmo 197

Criar um algoritmo que leia um número que servirá para controlar os primeiros números ímpares. Deverá ser impressa a soma desses números. Suponha que num será maior que zero.

Quantos: 5 *Saída: 25*
(1 3 5 7 9) – primeiros ímpares

```
prog para24
 int num,impar,i;
 imprima "\nDigite um numero maior que zero: ";
 leia num;
 impar <-0;
 para( i <- 1; i <= num * 2  ;  i <- i + 2)
 {   impar <-impar + i; }
 imprima "\n", impar;
 imprima "\n";
fimprog
```

algoritmo 198

Criar um algoritmo que leia os limites inferior e superior de um intervalo e imprima todos os números naturais no intervalo fechado. Suponha que os dados digitados são para um intervalo crescente. Exemplo:

Limite inferior: 5 *Saída: 5 6 7 8 9 10 11 12*
Limite superior: 12

```
prog para25
 int ini, f, i;
 imprima "\nvalor inicial e final, pressionando enter apos digitar cada
um: ";
 leia ini;
 leia f;
 para( i <- ini; i <= f;  i ++)
 {   imprima i, "    "; }
 imprima "\n";
fimprog
```

algoritmo 199

Criar um algoritmo que leia os limites inferior e superior de um intervalo e imprima todos os números múltiplos de 6 no intervalo fechado. Suponha que os dados digitados são para um intervalo crescente. Exemplo:

Limite inferior: 5 *Saída: 6 12*
Limite superior: 13

```
prog para26
 int ini, f, i;
 imprima " digite valor inicial e valor final de um intervalo,
pressionando enter apos digitar cada um: ";
 leia ini;
 leia f;
 se( ini % 6 == 0)
 {  ini <- ini  +  6;  }
 senao
 {  ini <- ini  +  (6 - ( ini % 6 )); }
 f--;
 para( i <- ini; i <= f;  i <- i  + 6 )
 { imprima i, "   "; }
 imprima "\n";
fimprog
```

algoritmo 200

Criar um algoritmo que leia os limites inferior e superior de um intervalo e o número cujos múltiplos se deseja que sejam impressos no intervalo aberto. Suponha que os dados digitados são para um intervalo crescente. Exemplo:

Limite inferior: 3 *Saída: 6 9*
Limite superior: 12
Número: 3

```
prog para27
 int ini, f, num, i;
 imprima "\ndigite valor inicial, valor final de um intervalo e o numero de
que se procuram os multiplos, pressionando enter apos digitar cada um: ";
 leia ini;
 leia f;
 leia  num;
 se( ini % num == 0)
 { ini <- ini  +  num; }
 senao
 { ini <- ini  + (num - ( ini % num ) );}
 f--;
 para( i <- ini; i <= f; i<- i + num )
 {  imprima i, "   ";}
 imprima "\n";
fimprog
```

algoritmo 201

Criar um algoritmo que leia os limites inferior e superior de um intervalo e imprima todos os números pares no intervalo aberto e seu somatório. Suponha que os dados digitados são para um intervalo crescente. Exemplo:

Limite inferior: 3 *Saída: 4 6 8 10*
Limite superior: 12 *Soma: 28*
Número: 3

```
prog para28
 int ini, vf, soma, i;
 imprima "\ndigite valor inicial e valor final de um intervalo,
          pressionando enter apos digitar cada um: ";
 leia ini;
 leia vf;
 soma <-0;
 se( ini % 2 == 0)
 { ini <- ini +  2; }
 senao
 { ini <- ini +  1; }
 vf--;
 para( i <- ini ; i <= vf;  i <- i +  2)
 {
    soma <- soma + i;
    imprima i, "   ";
 }
 imprima "\nsoma = ", soma;
 imprima "\n";
fimprog
```

algoritmo 202

Criar um algoritmo que leia um número (num) da entrada e imprima os múltiplos de 3 e 5 ao mesmo tempo no intervalo de 1 a num. Exemplo:

Numero lido: 50 *Saida: 15 30 45*

```
prog para29
 int  i, num;
 imprima "\nEntre com um numero maior do que 15: ";
 leia num;
 se(num>=15)
 {
   para( i <- 15 ; i <= num;  i <- i + 15 )
   { imprima i, "   "; }
 }
 senao
 { imprima "\n NAO EXISTE";}
 imprima "\n";
fimprog
```

algoritmo 203

Criar um algoritmo que leia um número (num) da entrada. Em seguida, ler n números da entrada e imprimir o triplo de cada um. Exemplo:

1º valor lido

5

5 valores lidos	*valores impressos*
3	*9*
10	*30*
12	*36*
2	*6*
1	*3*

```
prog para30
 int i, num;
 real num1;
 imprima "\nQuantos numeros voce quer digitar? ";
 leia num;
 para( i <- 1 ; i <= num;  i ++ )
 {
    imprima "\nDigite o ", i ," numero de ", num, ": ";
    leia num1;
    imprima "\n", num1 * 3;
 }
 imprima "\n";
fimprog
```

algoritmo 204

Criar um algoritmo que leia um número da entrada (num), leia n números inteiros da entrada e imprima o maior deles. Suponha que todos os números lidos serão positivos. Exemplo:

1º valor lido

3

3 valores lidos	*Valor impresso*
3	*18*
18	
12	

```
prog para31
 int i, num;
 real num1, maior;
 imprima "\nQuantos numeros voce quer digitar? ";
 leia num;
```

```
 maior<--1.;
 para( i <- 1; i <=num; i++)
 {
  imprima "\nDigite um numero: ";
  leia num1;
  se(num1 > maior)
  {  maior <-num1; }
 }
 imprima "\nEste  o maior numero que voce digitou: ", maior;
 imprima "\n";
fimprog
```

algoritmo 205

Criar um algoritmo que leia um número da entrada (num), leia n números inteiros da entrada e imprima o maior deles. ***Não*** *suponha que todos os números lidos serão positivos. Exemplo:*

1º valor lido

3

3 valores lidos	*Valor impresso*
–7	*–3*
–2	
–15	

```
prog para32
 int i, num;
 real num1, maior;
 imprima "\nQuantos numeros voce quer digitar? ";
 leia num;
 imprima "\nDigite um numero: ";
 leia num1;
 maior<- num1;
 para( i <- 1; i <= num - 1; i++)
 {
  imprima "Digite um numero: ";
  leia num1;
  se(num1 > maior)
  { maior <-num1; }
 }
 imprima "\nMaior numero digitado: ", maior;
 imprima "\n";
fimprog
```

algoritmo 206

Criar um algoritmo que leia a quantidade de números que se deseja digitar para que possa ser impresso o maior e o menor número digitados. ***Não*** *suponha que todos os números lidos serão positivos.*

```
prog para33
 int i, quant;
 real num, maior, menor;
 imprima "Quantos numeros voce quer digitar? ";
 leia quant;
 imprima "\nEntre com um numero: ";
 leia num;
 maior <- num;
 menor <- num;
 para( i <- 1; i < quant; i++)
 {
   imprima "\nEntre com um numero: ";
   leia num;
   se(num > maior)
   {  maior <- num;}
   senao
   {
    se(num < menor)
    { menor <- num;}
   }
 }
 imprima "\nMaior: ", maior;
 imprima "\nMenor: ", menor;
 imprima "\n";
fimprog
```

algoritmo 207

Sabendo-se que a UAL calcula o produto através de somas sucessivas, criar um algoritmo que calcule o produto de dois números inteiros lidos. Suponha que os números lidos sejam positivos e que o multiplicando seja menor do que o multiplicador.

```
prog para34
 int num1, num2, i, soma ;
 imprima "\nentre com o multiplicando: ";
 leia num1;
 imprima "\nentre com o multiplicador: ";
 leia num2 ;
 soma <-0;
 para( i <- 1; i <= num2; i++)
 { soma <- soma + num1;}
```

```
 imprima "\nProduto: ", soma;
 imprima "\n";
fimprog
```

algoritmo 208

Criar um algoritmo que imprima os 10 primeiros termos da série de Fibonacci.

Observação: os dois primeiros termos desta série são 1 e 1 e os demais são gerados a partir da soma dos anteriores. Exemplos:

- *1 + 1 -> 2 terceiro termo;*
- *1 + 2 -> 3 quarto termo etc.*

```
prog para35
 int i,ant,atual,prox;
 ant<- 0;
 atual<-1;
 para( i <- 1; i <=10; i++)
 {
   imprima "\n", atual,"  ";
   prox <- ant +atual;
   ant <- atual;
   atual <- prox;
 }
 imprima "\n";
fimprog
```

algoritmo 209

A série de RICCI difere da série de FIBONACCI porque os dois primeiros termos são fornecidos pelo usuário. Os demais termos são gerados da mesma forma que a série de FIBONACCI. Criar um algoritmo que imprima os n primeiros termos da série de RICCI e a soma dos termos impressos, sabendo-se que para existir esta série serão necessários pelo menos três termos.

```
prog para36
 int a1, a2, i, n, termo, soma;
 imprima "\nEntre com 1 termo: ";
 leia a1;
 imprima "\nEntre com 2 termo: ";
 leia a2;
 imprima "\nEntre com n termos: ";
 leia n;
 soma <- a1 + a2;
 se(n >= 3)
 {
  imprima "\n", a1, "  " ,a2, "  ";
  para( i <- 1; i <= n - 2; i++)
```

```
  {
     termo  <- a1 + a2;
     a1  <- a2;
     a2  <-  termo;
     imprima termo, "  ";
     soma <- soma + termo;
    }
    imprima "\nSoma: ", soma;
 }
 senao
 {  imprima "\nnao tem serie"; }
 imprima "\n";
fimprog
```

algoritmo 210

A série de FETUCCINE é gerada da seguinte forma: os dois primeiros termos são fornecidos pelo usuário; a partir daí, os termos são gerados com a soma ou subtração dos dois termos anteriores, ou seja:

$A_I = A_{I-1} + A_{I-2}$ *para I ímpar*
$A_I = A_{I-1} - A_{I-2}$ *para I par*

Criar um algoritmo que imprima os 10 primeiros termos da série de FETUCCINE.

```
prog para37
 int a1, a2, i, termo;
 imprima "\nEntre com 1 termo: ";
 leia a1;
 imprima "\nEntre com 2 termo: ";
 leia a2;
 imprima "\n", a1, "  " ,a2, "  ";
 para( i <- 3; i <= 10; i++)
 {
   se( i % 2 == 0)
   {   termo <- a2 - a1; }
   senao
   {  termo <- a2 + a1; }
   imprima termo, "  ";
   a1 <- a2;
   a2 <- termo;
 }
 imprima "\n";
fimprog
```

algoritmo 211

Criar um algoritmo que imprima todos os números inteiros e positivos no intervalo aberto entre 10 e 100 de modo que:

- *não terminem em zero;*
- *se o dígito da direita for removido, o número restante é divisor do número original*

Exemplos:

26: 2 é divisor de 26
80: 8 é divisor de 80

```
prog para38
 int num , aux;
 para( num <- 11; num <= 99; num++)
 {
  se(num % 10 < >  0 )
  {
    aux <- num div 10;
    se( num % aux == 0)
    {  imprima num, "   "; }
  }
 }
 imprima "\n";
fimprog
```

algoritmo 212

Entrar com 20 números e imprimir a soma dos números cujos quadrados são menores do que 225.

```
prog algoritmo212
 int a;
 real numero,soma;
 soma <- 0.;
 para(a <- 1; a <=20; a++)
 {
   imprima "\nEntre com o numero: ";leia numero;
   se(numero ** 2 < 225.)
   { soma <- soma + numero; }
 }
 imprima  "\nsoma: ",soma;
 imprima "\n";
fimprog
```

algoritmo 213

Entrar com 12 números e imprimir a média desses números.

```
prog para40
 int a;
 real numero, soma;
 soma <- 0. ;
 para( a <- 1; a <=12; a++)
```

```
 {
  imprima "\ndigite numero: ";
  leia numero;
 soma  <-soma + numero;
 }
 imprima  "\nmedia: ", soma/12 ;
 imprima "\n";
fimprog
```

algoritmo 214

Entrar com nome, nota da PR1 e nota da PR2 de 15 alunos. Imprimir uma listagem, contendo: nome, nota da PR1, nota da PR2 e média arredondada de cada aluno. Ao final, calcule a média geral da turma.

```
prog para41
 int c;
 string nome;
 real pr1, pr2, soma, media;
 soma <- 0. ;
 imprima "\nRELACAO DOS ALUNOS";
 para( c <- 1; c <=15; c++)
 {
   imprima "\n\ndigite nome: ";
   leia nome;
   imprima "\ndigite nota 1: ";
   leia pr1;
   imprima "\ndigite nota 2: ";
   leia pr2;
   media  <- (pr1 + pr2)/2;
   soma   <- soma + media;
imprima "\n",nome,"   ", pr1,"   ", pr2,"   ", realint(media + 0.0001) ;
 }
 imprima  "\nmedia: ", soma / 15 ;
 imprima "\n";
fimprog
```

algoritmo 215

Entrar com um número e imprimir todos os seus divisores.

```
prog para421
 int c, num;
 imprima "\ndigite numero: ";
 leia num;
 para( c <- 1; c <=num; c++)
 {
   se( num  % c == 0 )
```

```
  {    imprima c, "  " ;    }
 }
 imprima "\n";
fimprog

prog para422
 /*melhor solucao para nao ficar procurando divisores depois da metade*/
 int c, num, vf;
 imprima "\ndigite numero: ";
 leia num;
 vf <- num  div 2;
 para( c <- 1; c <= vf; c++)
 {
   se( num  % c == 0 )
  {   imprima c,"  " ;  }
 }
 imprima  num;
 imprima "\n";
fimprog
```

algoritmo 216

Ler 200 números inteiros e imprimir quantos são pares e quantos são ímpares.

```
prog para43
 int numero, cp, c;
 cp <- 0;
 para( c <- 1; c <=200; c++)
 {
   imprima "\ndigite numero: ";
   leia numero;
   se( numero % 2 == 0 )
   {   cp <-  cp +1;  }
 }
 imprima "\npares: ", cp;
 imprima "\nimpares: ", 200-cp;
 imprima "\n";
fimprog
```

algoritmo 217

Entrar com oito nomes e imprimir quantas letras tem cada nome.

```
prog para44
 int a;
 string nome;
 para( a <- 1; a <=8; a++)
 {
  imprima "\ndigite nome: ";
```

```
  leia nome;
  imprima  "\nnumero de letras: ", strtam( nome) ;
 }
 imprima "\n";
fimprog
```

algoritmo 218

Entrar com 12 nomes e imprimir o primeiro caractere de cada nome.

```
prog para45
 int a;
 string nome;
 para( a <- 1; a <=12; a++)
 {
   imprima "\ndigite nome: ";
   leia nome;
   imprima  "\n1a letra: ", strprim(nome) ;
 }
 imprima "\n";
fimprog
```

algoritmo 219

Entrar com o número de vezes que se deseja imprimir a palavra SOL e imprimir.

```
prog para46
 int c, num, vf;
 imprima "\ndigite numero de vezes que deseja imprimir a palavra SOL? ";
 leia num;
 para( c <- 1; c <=num; c++)
 {  imprima  "SOL " ; }
 imprima "\n";
fimprog
```

algoritmo 220

Entrar com um nome e imprimi-lo tantas vezes quantos forem seus caracteres.

```
prog para47
 int c, vf;
 string nome;
 imprima "\ndigite nome: ";
 leia nome;
 vf <- strtam(nome);
 para( c <- 1; c <=vf; c++)
 {   imprima "\n",nome ; }
 imprima "\n";
fimprog
```

algoritmo 221

Criar um algoritmo que entre com uma palavra e imprima conforme o exemplo a seguir:

palavra: PAZ

P
A
Z

```
prog para48
 int c, vf;
 string nome;
 imprima "\ndigite nome: ";
 leia nome;
 vf <- strtam(nome) -  1 ;
 para( c <- 0; c <= vf; c++)
 {    imprima "\n", strelem(nome, c) ;  }
 imprima "\n";
fimprog
```

algoritmo 222

Criar um algoritmo que entre com uma palavra e imprima conforme o exemplo a seguir:

palavra: PAZ

ZAP

```
prog para49
 int c, vf;
 string nome;
 imprima "\ndigite nome: ";
 leia nome;
 vf <- strtam(nome) -  1;
 para( c <- vf; c >= 0; c–)
 {   imprima  strelem(nome, c) ;  }
 imprima "\n";
fimprog
```

algoritmo 223

Criar um algoritmo que entre com uma palavra e imprima conforme o exemplo a seguir:

palavra: AMOR

A
AM
AMO
AMOR

```
prog para50
 int a;
 string nome;
 imprima "\ndigite nome: ";
 leia nome;
 para(a <- 1;a <= strtam(nome);a++)
 {  imprima "\n",strnprim(nome,a); }
 imprima "\n";
fimprog
```

algoritmo 224

Criar um algoritmo que entre com uma palavra e imprima conforme o exemplo a seguir:

palavra: AMOR

> ***AMOR***
>
> ***AMO***
>
> ***AM***
>
> ***A***

```
prog para51
 int a;
 string nome;
 imprima "\ndigite nome: ";
 leia nome;
 para(a <- strtam(nome);a >= 1;a–)
 { imprima "\n",strnprim(nome,a); }
 imprima "\n";
fimprog
```

algoritmo 225

Criar um algoritmo que entre com uma palavra e imprima conforme o exemplo a seguir:

palavra: TERRA

> ***A***
>
> ***RA***
>
> ***RRA***
>
> ***ERRA***
>
> ***TERRA***

```
prog para52
 int a;
 string nome;
```

```
 imprima "\ndigite nome: ";
 leia nome;
 para(a <-strtam(nome) - 1; a >= 0 ; a–)
 {  imprima "\n",strnresto(nome,a); }
 imprima "\n";
fimprog
```

algoritmo 226

Criar um algoritmo que entre com uma palavra e imprima conforme o exemplo a seguir:

palavra: TERRA

TERRA

ERRA

RRA

RA

A

```
prog para53
 int a;
 string nome;
 imprima "\ndigite nome: ";
 leia nome;
 para(a <- 0; a <= strtam(nome) - 1;a++)
 {  imprima "\n",strnresto(nome,a);}
 imprima "\n";
fimprog
```

algoritmo 227

Criar um algoritmo que entre com uma palavra e se a palavra tiver número ímpar de caracteres, imprima conforme o exemplo a seguir; caso contrário, imprima: NÃO FACO

palavra: SONHO

SONHO

ONHO

N

```
prog para54
 int c, c1, L,tam,t1;
 string nome;
 imprima "\ndigite nome: ";
 leia nome;
 L <- strtam(nome);
```

```
 se( L % 2 < > 0 )
 { tam <-strtam(nome);
   para( c <- 0; c <=tam div 2 ; c++)
   {
       imprima strnresto(nome, c), "\n";
       t1   <- strtam(nome) - 1 ;
       nome <- strnprim(nome,t1);
       para(c1 <-1; c1<=c+1; c1++)
       {imprima " "; }
    }
 }
 senao
 {  imprima "\nNAO FACO "; }
 imprima "\n";
fimprog
```

algoritmo 228

Criar um algoritmo que entre com uma palavra e se a palavra tiver número ímpar de caracteres, imprima conforme o exemplo a seguir; caso contrário, imprima: NÃO FACO

palavra: SONHO

N

ONH

SONHO

```
prog para55
 int c,c1,tam,meio, L,p1, p2,t1;
 string nome, n1, n2,n;
 imprima "\ndigite nome: ";
 leia nome;
 L <- strtam(nome);
 se( L % 2 < > 0 )
 { meio <-L div 2 ;
   tam <-meio;
   p1  <-meio;
   p2  <-meio;
   n   <-strelem(nome,p1);
   para(c1 <-1; c1<=meio; c1++)
   { imprima " "; } meio<-meio-1;
   imprima n, "\n";
   para( c <- 1; c <= tam ; c++)
   {   para(c1 <-1; c1<=meio; c1++)
      {imprima " "; } meio<-meio-1;
      p1 <- p1 - 1; # p1--;
```

algoritmo 233

Entrar com dois números e imprimir todos os números no intervalo fechado, do menor para o maior.

```
prog para60
 int ini, f, aux, i;
 imprima " digite limite inferior: ";
 leia ini;
 imprima "\ndigite limite supeior: ";
 leia f;
 se( ini > f )
 {
   aux <- ini ;
   ini <- f ;
   f <- aux ;
 }
 para( i <- ini; i <= f; i++ )
 {   imprima i, "    ";}
 imprima "\n";
fimprog
```

algoritmo 234

Entrar com nome e salário bruto de 10 pessoas. Imprimir nome e o valor da alíquota do imposto de renda:

salário menor que R$ 600,00	*– isento*
salário >= R$ 600,00 e < R$ 1500,00	*– 10% do salário bruto*
salário >= R$ 1500,00	*– 15% do salário bruto*

```
prog para61
 int i;
 string nome;
 real salario, ir;
 para( i <- 1; i <= 20; i++ )
 {
  imprima "\ndigite nome: ";
  leia nome;
  imprima "\ndigite salario: ";
  leia salario;
  se( salario <600.)
  {  ir<- 0.;  }
  senao
  {
    se( salario <1500.)
    {    ir<- salario * 0.1;    }
```

```
    senao
    {   ir<- salario *0.15;  }
  }
  imprima "\n", nome, "  R$ ", formatar(salario,2), "bR$b",
formatar(ir,2);
 }
 imprima "\n";
fimprog
```

algoritmo 235

Entrar com dez números (positivos ou negativos) e imprimir o maior e o menor número da lista.

```
prog para62
 int i, num;
 real num1, maior, menor;
 imprima "Digite um numero: ";
 leia num1;
 maior <- num1;
 menor <- num1;
 para( i <- 1; i <=9; i++)
 {
    imprima "Digite um numero: ";
    leia num1;
    se(num1 > maior)
    { maior <-num1; }
    senao
    {
       se(num1 < menor)
       { menor <-num1; }
     }
 }
 imprima "\nMaior numero digitado: ", maior;
 imprima "\nMenor numero digitado: ", menor;
 imprima "\n";
fimprog
```

algoritmo 236

Ler o número de termos da série (n) e imprimir o valor de H, sendo:

$$H = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{N}$$

```
prog para63
 int c, num;
 real h;
 h <- 0. ;
```

```
 imprima "\ndigite numero: ";
 leia num;
 para( c<- 1; c  <= num; c++)
 {   h <- h + 1 / c; }
 imprima "\n soma: ", h ;
 imprima "\n";
fimprog
```

algoritmo 237

Ler o número de termos da série (n) e imprimir o valor de H, sendo:

$$H = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + ... + \frac{1}{N}$$

```
prog para64sol1
 int c, num;
 real h;
 h <- 0. ;
 imprima "\ndigite numero: ";
 leia num;
 imprima "\nPrimeira Solucao ";
 para( c<- 1; c  <= num; c++)
 {
   se( c % 2 == 1 )
   {   h <- h + 1 / c;  }
   senao
   {   h <- h - 1 / c;  }
 }
 imprima "\nsoma: ", h ;
 imprima "\n";
fimprog

prog para64sol2
 int c, num;
 real h;
 h <- 0. ;
 imprima "\ndigite numero: ";
 leia num;
 imprima "\nSegunda Solucao ";
 para( c<- 1; c  <= num; c++)
 {    h <- h + ( -1**(c+1)) / c; }
 imprima "\n\nsoma: ", h ;
 imprima "\n";
fimprog

prog para64sol3
 int c, num, sinal;
```

```
 real h;
 h <- 0. ;
 sinal <- 1;
 imprima "\ndigite numero: ";
 leia num;
 imprima "\nTerceira Solucao ";
 para( c<- 1; c  <= num; c++)
 {   h <- h + sinal * 1/c; sinal <- sinal * (-1); }
 imprima "\n\nsoma: ", h ;
 imprima "\n";
fimprog
```

algoritmo 238

Ler o número de termos da série (N) e imprimir o valor de S, sendo :

$$S = \frac{1}{N} + \frac{1}{N-1} + \frac{3}{N-2} + ... + \frac{N-1}{2} + N$$

```
prog para65
 int c, num, n;
 real h;
 h <- 0. ;
 imprima "\ndigite numero: ";
 leia num;
 n <- num;
 para( c<- 1; c  <= num; c++)
 {
  h <- h + c / n;
  n <- n -1;
 }
 imprima "\nsoma: ", h ;
 imprima "\n";
fimprog
```

algoritmo 239

Implementar um algoritmo para calcular o valor de e^X. O valor de X deverá ser digitado. O valor de e^X será calculado pela soma dos 10 primeiros termos da série a seguir:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + ...$$

sabe-se que: 0! é igual a 1

```
prog para66
 int c, fat;
 real x,soma;
 soma <- 1.; # 0! = 1
```

```
 fat <- 1;
 imprima "\ndigite valor de x: ";
 leia x;
 para( c <- 1; c <= 9; c++)
 {
   fat <-  fat * c;
   soma <-  soma + x ** c / fat;
 }
 imprima "\nvalor de e elevado a ", X, ": ", soma;
 imprima "\n";
fimprog
```

algoritmo 240

Implementar um algoritmo para calcular o sen(X). O valor de X deverá ser digitado em graus. O valor do seno de X será calculado pela soma dos 10 primeiros termos da série a seguir:

$$sen\ x = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \ldots$$

```
prog  para67
 int c,d, expden, fat;
 real x,seno;
 seno <- 0.;
 expden <- 1;
 imprima "\ndigite valor de x: ";
 leia x;
 x <- x * pi/180;
 para( c <- 1; c<= 10;c++)
 { fat <- 1;
   para( d <- 1; d <= expden; d++)
  { fat <- fat * d; }
   se(c % 2 == 0)
  { seno <- seno- x ** expden / fat;}
   senao
  {  seno <- seno + x ** expden / fat;}
   expden <- expden + 2;
 }
 imprima  "\nseno: ", formatar(seno,4);
 imprima "\n";
fimprog
```

algoritmo 241

Criar um algoritmo que imprima a tabuada de multiplicar do número 3.

```
prog para68
 int c;
```

```
 imprima "\nTABUADA DO 3\n";
 para(c<-1; c<=10; c++)
 {   imprima "\n3", "ƀxƀ", c, "ƀ=ƀ", 3*c;    }
 imprima "\n";
fimprog
```

algoritmo 242

Criar um algoritmo que deixe escolher qual a tabuada de multiplicar que se deseja imprimir.

```
prog para69
 int c, n;
 imprima "\n qual tabuada? ";
 leia n;
 se( n > 0)
 {
   imprima "\nTABUADA DOƀ", n,"\n";
   para(c<-1; c<=10; c++)
   { imprima "\n",n, "ƀxƀ", c, "ƀ=ƀ", n*c; }
 }
 senao
 { imprima "\n Nao existe tabuada";  }
 imprima "\n";
fimprog
```

algoritmo 243

Entrar com uma mensagem e imprimir quantas letras A, E, I, O e U tem esta mensagem (considerar minúscula e maiúscula).

```
prog para70
 int L,ca,ce,ci,co,cu;
 string msg,letra;
 ca <-0;
 ce<-0;
 ci <-0;
 co<-0;
 cu<-0;
 imprima "\nDigite uma mensagem: ";
 leia msg;
 para( L <- 0; L <strtam(msg); L++)
 {
   letra<-strelem(msg,L);
   se(letra=="a" || letra =="A")
   {ca++;}
   senao
   {
     se(letra=="e" || letra =="E")
```

```
      {ce++;}
      senao
      {
        se(letra=="i" || letra =="I")
        {ci++;}
        senao
        {
          se(letra=="o" || letra =="O")
          {co++;}
          senao
          {
            se(letra=="u" || letra =="U")
            {cu++;}
          }
        }
      }
    }
  }
 imprima "\n Total de letras A: ", ca;
 imprima "\n Total de letras E: ", ce;
 imprima "\n Total de letras I: ", ci;
 imprima "\n Total de letras O: ", co;
 imprima "\n Total de letras U: ", cu;
 imprima "\n";
fimprog
```

algoritmo 244

Entrar com uma mensagem e criptografá-la da seguinte maneira:

A – X ; E – Y ; I – W ; O – K ; U – Z

```
prog para71
 int L;
 string msg,letra;
 imprima "\nDigite uma mensagem: ";
 leia msg;
 para( L <- 0; L <strtam(msg); L++)
 {
   letra<-strelem(msg,L);
   se(letra=="a" || letra =="A")
   {imprima "X";}
   senao
   {
     se(letra=="e" || letra =="E")
     {imprima "Y";}
     senao
     {
       se(letra=="i" || letra =="I")
```

```
        {imprima "W";}
        senao
        {
          se(letra=="o" || letra =="O")
          {imprima "K";}
          senao
          {
            se(letra=="u" || letra =="U")
            {imprima "K";}
            senao
            {imprima letra;}
          }
        }
      }
    }
  }
  imprima "\n";
fimprog
```

algoritmo 245

Criar um algoritmo que receba a idade e o peso de 20 pessoas. Calcular e imprimir as médias dos pesos das pessoas da mesma faixa etária. As faixas etárias são: de 1 a 10 anos, de 11 a 20 anos, de 21 a 30 anos e maiores de 30 anos.

```
prog  para72
 int L, idade, c1,c2,c3,c4;
 real peso,p1,p2,p3,p4;
 p1 <-0.;
 p2<-0.;
 p3 <-0.;
 p4 <-0.;
 c1<-0;
 c2<-0;
 c3<-0;
 c4<-0;
 para( L <- 1; L <=20; L++)
 {
  imprima "\nDigite idade: ";
  leia idade;
  imprima "\nDigite peso: ";
  leia peso;
  se( idade<=10)
  {c1++; p1 <- p1  + peso;}
  senao
  {
    se( idade<=20)
    {c2++; p2 <- p2  + peso;}
```

```
    senao
    {
      se( idade<=30)
      {c3++; p3 <- p3  + peso;}
      senao
      {c4++; p4 <- p4  + peso;}
     }
   }
  }
 se(c1< >0)
 {imprima "\nMedia dos pesos 1-10: ", p1/c1;}
 senao
 {imprima"\nNinguem com idades de 1-10";}
 se(c2< >0)
 {imprima "\nMedia dos pesos 11-20: ", p2/c2;}
 senao
 {imprima"\nNinguem com idades de 11-20";}
 se(c3< >0)
 {imprima "\nMedia dos pesos 21-30: ", p3/c3;}
 senao
 {imprima"\nNinguem com idades de 21-30";}
 se(c4< >0)
 {imprima "\nMedia dos pesos maiores que 30: ", p4/c4;}
 senao
 {imprima"\nNinguem com idades maiores que 30";}
 imprima "\n";
fimprog
```

algoritmo 246

No dia da estréia do filme "Senhor dos Anéis", uma grande emissora de TV realizou uma pesquisa logo após o encerramento do filme. Cada espectador respondeu a um questionário no qual constava sua idade e a sua opinião em relação ao filme: excelente – 3; bom – 2; regular – 1. Criar um algoritmo que receba a idade e a opinião de 20 espectadores, calcule e imprima:

- *a média das idades das pessoas que responderam excelente;*
- *a quantidade de pessoas que responderam regular;*
- *a percentagem de pessoas que responderam bom entre todos os expectadores analisados.*

```
prog para73
 int  idade, op, I, opexc, opbom, opregular, idadegot;
 idadegot  <-  0;
 opexc  <-  0;
 opbom  <-  0;
 opregular  <-  0;
 para( I <- 1; I <=15; I++)
 { imprima "\n Digite idade: " leia idade;
```

```
imprima "\n Digite  opiniao: excelente - 3; bom - 2; regular - 1: ";
leia op ;
se( op == 3 )
{ opexc++;
 idadegot  <-  idadegot + idade;}
senao
 {  se( op == 1 )
    { opregular ++;}
    senao
    {  se( op ==2 )
       { opbom ++;}
    }
  }
}
se(opexc<>0)
{imprima "\nMedia de idade das pessoas que responderam excelente: ",
           idadegot/ opexc;}
senao
{imprima "\nNinguem que tenha escolhido excelente";}
se(opregular<>0)
{imprima "\nQuantidade de pessoas que responderam regular: ", opregular ;}
senao
{imprima "\nNinguem que tenha escolhido regular";}
se(opbom<>0)
{imprima "\nA porcentagem das pessoas que respondeu Bom: ",  opbom * 100 /15 ;}
senao
{imprima "\nNinguem que tenha escolhido bom";}
imprima "\n";
fimprog
```

algoritmo 247

Num campeonato europeu de volleyball, se inscreveram 30 países. Sabendo-se que na lista oficial de cada país consta, além de outros dados, peso e idade de 12 jogadores, criar um algoritmo que apresente as seguintes informações:

- *o peso médio e a idade média de cada um dos times;*
- *o peso médio e a idade média de todos os participantes.*

```
prog para74
 int i, x, id, somaid, totid;
 real peso, totpeso, somapeso;
 totpeso <- 0.;
 totid <- 0;
 para( i <- 1; i <= 30; i++)
 { somapeso <- 0.;
   somaid <- 0;
   para( x <- 1; x <= 12; x++)
   {  imprima "\nDigite peso: ";
     leia peso;
```

```
      imprima "\nDigite idade: ";
      leia id;
      somapeso <- somapeso + peso;
      somaid <- somaid + id;
    }
    imprima "\npeso medio do time: ", somapeso/ 12;
    imprima "\nidade media do time: ", somaid/ 12;
    totpeso <-  totpeso + somapeso;
    totid <- totid + somaid;
  }
  imprima "\npeso medio de todos os times: ", totpeso / 360;
  imprima "\nidade media de todos os times: ", totid / 360;
  imprima "\n";
fimprog
```

algoritmo 248

Criar um algoritmo que entre com uma palavra e imprima conforme exemplo a seguir:

palavra: SONHO

SONHO

SONHO SONHO

SONHO SONHO SONHO

SONHO SONHO SONHO SONHO

SONHO SONHO SONHO SONHO SONHO

```
prog para75
 int c, c1, L;
 string nome;
 imprima "\ndigite nome: ";
 leia nome;
 L <- strtam(nome);
 para( c <- 1; c <=L ; c++)
 {
   para(c1<-1; c1<=c; c1++)
   {   imprima nome, "ƀ";   }
   imprima "\n";
 }
 imprima "\n";
fimprog
```

algoritmo 249

Criar um algoritmo que entre com uma palavra e se a palavra tiver número ímpar de caracteres, imprima conforme exemplo a seguir, caso contrário, imprima : NAO FACO:

palavra: SONHO

SONHO

SONHO SONHO SONHO

SONHO SONHO SONHO SONHO SONHO

```
prog para76
 int c, c1, L;
 string nome;
 imprima "\ndigite nome: ";
 leia nome;
 L <- strtam(nome);
 se( L % 2 < >0)
 {
   para( c <- 1; c <=L ; c<-c+2)
   {
      para(c1<-1; c1<=c; c1++)
      {   imprima nome, "ƀ";    }
      imprima "\n";
   }
  }
 senao
 {  imprima "\nNao faco";}
 imprima "\n";
fimprog
```

algoritmo 250

Imprimir todas as tabuadas de multiplicar de 1 até 10.

```
prog para77
 int L, c;
 string r;
 para( L <- 1; L <=10 ; L++)
 {
   imprima "\nTABUADA DO ", L, "\n";
   para(c<-1; c<=10; c++)
   {  imprima "\n",L, "  x  ", c, " = ", L*c;  }
   imprima "\n\nPressione enter: ";
   leia r;
 }
 imprima "\n";
fimprog
```

algoritmo 251

Criar um algoritmo para imprimir uma tabela para DEZ times num torneio de rodada dupla.

```
prog para78
 int L, c;
 string r;
 para( L <- 1; L <=10 ; L++)
 {
   para(c<-1; c<=10; c++)
   {
      se(L < > c)
      {   imprima "\ntime : ",L, "ƀxƀtime : ", c, " ________"; }
   }
   imprima "\n\nPressione enter: ";
   leia r;
 }
 imprima "\n";
fimprog
```

algoritmo 252

Criar um algoritmo para imprimir uma tabela para DEZ times num torneio de rodada simples.

```
prog para79
 int L, c;
 string r;
 para( L <- 1; L <=9 ; L++)
 {
   para(c<-L+1; c<=10; c++)
   {  imprima "\ntime : ",L, "ƀxƀtime : ", c, " ________"; }
   imprima "\n\nPressione enter: ";
   leia r;
  }
 imprima "\n";
fimprog
```

algoritmo 253

Criar um algoritmo que entre com dez mensagens, e, para cada mensagem, imprimir quantas letras A tem (considerar maiúsculas e minúsculas).

```
prog para80
 int L,c, ca;
 string msg,letra;
 para(L<-1; L<=10; L++)
 {
```

```
  ca <-0;
  imprima "\nDigite uma mensagem: ";
  leia msg;
  para( c <- 0; c <strtam(msg); c++)
  {
     letra<-strelem(msg,c);
     se(letra=="a" || letra =="A")
     {ca ++;}
   }
   imprima "\n Total de letras A: ", ca;
 }
 imprima "\n";
fimprog
```

algoritmo 254

Criar um algoritmo que entre com dez notas de cada aluno de uma turma de 20 alunos e imprima:

- *a média de cada aluno*
- *a média da turma*
- *o percentual de alunos que tiveram médias maiores ou iguais a 5.0.*

```
prog para81
 int L,c, cap;
 real somaa, somat, nota, media;
 somat <- 0.;
 cap <- 0;
 para(L <- 1;L <= 20;L++)
 {
    somaa  <-  0.;
    para(c <- 1;c <= 10;c++)
    {
      imprima "\nDigite ", c,"ª nota do aluno ",L, ": ";
      leia nota;
      somaa <- somaa +nota;
   }
  media <- somaa/10;
  se(media>=5.)
  {cap++;}
  imprima "\nMedia do ", L,"º aluno: ",media;
  somat <- somat+media;
 }
 imprima "\nMedia da turma: ", somat/20;
 imprima "\nPercentual de Aprovados: ", cap*100/20;
 imprima "\n";
fimprog
```

algoritmo 255

Uma escola tem 5 turmas e cada turma tem n alunos. Criar um algoritmo que imprima, por turma, total de alunos com média superior a 7 e a média geral da escola.

```
prog para82
 int L,c, cap, n, totalunos;
 real media, somageral;
 somageral <- 0.;
 totalunos <- 0;
 para(L <- 1;L <= 5;L++)
 {
   cap <- 0;
   imprima "\nQuantos alunos tem a ", L, "ª turma? ";
   leia n;
   totalunos <- totalunos + n;
   para(c <- 1;c <= n;c++)
   {
     imprima "\nDigite media do ", c, "º  aluno: ";
     leia media;
      se(media >7.)
      {cap++;}
      somageral <- somageral + media;
    }
    imprima "\nTotal de alunos com media superior ou igual a 7: ",cap;
    imprima "\n";
  }
  imprima "\nMedia da escola: ", somageral / totalunos;
 imprima "\n";
fimprog
```

algoritmo 256

Palíndromos são palavras (frases também) que são iguais quando lidas de frente para trás e de trás para frente, ignorando os espaços.

AME O POEMA   AMOR A ROMA   ATE O POETA   LUZ AZUL

Criar um algoritmo que possa entrar com 15 mensagens e imprimir quantas são palíndromos.

```
prog  para83
 string PAL,PALSE, INV, LETRA ;
 int TAM, IND, C, CONT;
 CONT <- 0;
 para(C <- 1; C <= 15 ; C ++)
 {
  imprima "\nDigite uma Palavra: ";
  leia  PAL ;
```

```
    TAM <- strtam(PAL)- 1;
    PALSE <- "";
    para( IND <- TAM; IND >= 0; IND--)
    {
       LETRA<- strelem(PAL, IND);
       se(LETRA <> "ƀ")
       {PALSE <- strconcat(PALSE, LETRA);}
    }
    TAM <- strtam(PALSE)- 1;
    INV <- "";
    para( IND <- TAM; IND >= 0; IND--)
    {
       LETRA <- strelem(PALSE, IND);
       INV <- strconcat(INV, LETRA);
    }
    se( PALSE == INV)
    { CONT++; }
 }
 imprima "\nForam encontradas tantas palavras palindromos: ", CONT;
 imprima "\n";
fimprog
```

algoritmo 257

Criar um algoritmo que leia o valor de N, imprima a seqüência a seguir e o resultado.

$N! / 0^2! - (N-1)! / 2^2! + (N-2)! / 4^2! - (N-3)! / 6^2! + \ldots 0! / (2N^2)!$

Exemplo: N = 4
4! / 0! – 3! / 4! + 2! / 16! – 1! / 36! + 0! / 64!
SOMA: 22.5035

```
prog para84
int L, C, Y, Z,FAT, FAT1, S;
real  SOMA;
imprima "\ndigite o numero de termos: ";
leia Y;
SOMA <- 0.;
C <- 0;
S <- 1;
para( L <- Y; L >= 0; L--)
{
  FAT <- 1;
  para(Z <- 1; Z <=L; Z++)
  { FAT <- FAT * Z; }
  FAT1 <- 1;
  para(Z <- 1; Z <=C; Z++)
  { FAT1 <- FAT1 * Z; }
  se( S == 1)
```

```
        { imprima L, "! /", C ^ 2,"!  - ";
          SOMA <- SOMA + FAT /(FAT1 ^2);
          S <- 0;
        }
        senao
        { imprima L,"! /", C ^  2,"!  + ";
            SOMA <- SOMA - FAT / (FAT1 ^ 2);
            S  <-  1;
        }
        C  <-  C + 2;
  }
  imprima "\nSOMA: ",SOMA;
  imprima "\n";
fimprog
```

ESTRUTURA DO ENQUANTO

Estrutura recomendada quando o número de repetições for desconhecido, sendo necessária uma chave (um teste) para interromper a repetição.

É uma estrutura que testa a condição no início, e só repete se a condição for verdadeira.

Chave, teste ou flag. O que é isto?

Sendo redundante, para sermos bem claros, toda estrutura de repetição repete um conjunto de comandos várias vezes. A estrutura do **para** tem uma variável para controlar o número de repetições mas a estrutura do **enquanto** não agrega nenhuma variável contadora e, desta forma, a repetição não pararia nunca (é quando dizemos: *entrou em loop*). Para impedir que isto aconteça, é montada uma expressão condicional, igual à que você utilizou na estrutura do **se**, para interromper a repetição. Um número, um caractere ou uma palavra deverá ser utilizado na expressão e passado para o usuário para que ele possa digitar quando desejar parar.

Simula, com facilidade, a estrutura do **faca enquanto** desde que usemos um comando de atribuição e a estrutura do **para** desde que criemos uma variável que terá seu valor incrementado dentro da estrutura de repetição. **É a mais "poderosa" estrutura de repetição.**

A variável ou variáveis do teste deverão ter seu valor atribuído através de um comando de leitura ou de atribuição, **antes** da estrutura, e **dentro** da estrutura, na maioria das vezes, como **último comando antes do** }.

```
enquanto ( condição)
{
       bloco de comandos
}
```

Nos exemplos a seguir, mostraremos a importância da chave e da inicialização da variável que faz parte da expressão que controla a repetição.

algoritmo 258

```
prog enquanto1
 real num;
 enquanto( num  > 0.)
 { imprima "\ndigite numero: ";
   leia num;
   imprima "\ndobro do numero: ",
   num * 2; # continua
 }
 imprima "\n";
fimprog
```

Fail: Variável num não-inicializada.

Explicando:
A primeira tendência quando aprendemos a estrutura do enquanto é tentar montar o corpo da repetição da mesma forma que fazíamos com o para.

Mas tendo em vista que assumiremos que as variávies não são inicializadas automaticamente, teremos problema com a estrutura do enquanto.

Como fazer um teste com uma variável que não tem nenhum valor?

algoritmo 259

```
prog enquanto2
 real num;
 imprima "\ndigite numero: ";
 leia num;
 enquanto( num  >0.)
 { imprima "\n dobro do numero:
 ", num  *2; } # continua
 imprima "\n";
fimprog
```

digite numero: 8

dobro: 16
dobro: 16
dobro: 16
dobro: 16
dobro: 16
dobro: 16
dobro: 16
dobro: 16
dobro: 16
dobro: 16
dobro: 16
dobro: 16
dobro: 16

Observação: O algoritmo entra em loop

Explicando:

Tendo em vista que nossa experiência anterior falhou, nossa segunda tendência é deslocar a leitura para fora da estrutura.

Mas, desta vez, teremos outro problema: o algoritmo entra em loop *pois nunca mais atribuímos valor à variável num. É preciso interromper pressionando <ctrl><c>.*

algoritmo 260

```
prog enquanto3
 real num;
 imprima "\ndigite numero: ";
 leia num;
 enquanto( num  >0.)
 {
  imprima "\n dobro do numero:
  ", num  *2;
  imprima "\ndigite numero: ";
  leia num;
 }
 imprima "\n";
fimprog
```

```
digite numero: 8

dobro: 16
digite numero: 45

dobro: 90
digite numero: 0
```

Explicando:
Depois de várias tentativas, percebemos que precisaríamos de um comando de leitura para poder entrar na estrutura do enquanto e um para poder sair.

Resumindo:
Na prática, usamos a estrutura do enquanto das seguintes maneiras:

> ***leia** nome da variável presente na condição;*
> ***enquanto** (condição que inclui a variável do comando de leitura)*
> **{**
>
> *bloco de comandos*
>
> ***leia** nome da variável presente na condição;*
> **}**

ou

> *nome da variável presente na condição <-valor;*
> ***enquanto** (condição que inclui a variável do comando de leitura)*
> **{**
>
> *bloco de comandos*

nome da variável presente na condição <- *nome da variável presente na condição* < **um operador** > *valor;*
}

"LEMA"

Um comando leia (ou de atribuição) antes do enquanto para *entrar na repetição* e um comando leia (ou de atribuição) antes do }, para sair da repetição.

Simulando a estrutura do para

```
prog enqpara
 int a;
 a <- 1;
 enquanto (a <=10)
 { imprima "\n",a;
    a++;
 }
 imprima "\n";
fimprog
```

ESTRUTURA DO FACA ENQUANTO

Estrutura recomendada quando o número de repetições for desconhecido, sendo necessária uma chave (um teste) para interromper a repetição.

Sua diferença em relação ao enquanto é que ela testa ao final, significando que ela executa o trecho pelo menos uma vez.

Essa estrutura também precisa de uma chave para interromper a repetição.

Muitas pessoas não entendem por que existe esta estrutura se a estrutura do **enquanto** pode fazer o que ela faz. Na verdade, tem muito programador que nem faz uso dela e, quando faz, é em programas que usam menus:

```
faca
{
        bloco de comandos
}
enquanto (condição)
```

Simulando a estrutura do para

```
prog facaenqpara
 int a;
 a <- 1;
 faca
 { imprima "\n",a;
   a++;
 }
 enquanto (a <=10)
 imprima "\n";
fimprog
```

Exemplo: Criar um algoritmo que funcione através do menu a seguir:

> *MENU – um pouco de tudo*
>
> *A - Armazena na variavel menor e imprime o nome que tiver o menor numero de caracteres entre tres*
> *B - Gera e imprime uma nova palavra*
> *C - Calcula e imprime a raiz a quarta de um numero*
> *F - Finalizar*
> *OPCAO:*

Solução:

algoritmo 261

```
prog menu
 string  op,menor,p1,p2,p3,np;
 int n;
 real num;
 faca
 {
 imprima "\n MENU - um pouco de tudo";
 imprima "\n A - Armazena na variavel menor e imprime o nome que tiver o
           menor numero de caracteres entre tres" ; # continuacao
 imprima "\n B - Gera e imprime uma nova palavra";
 imprima "\n C - Calcula e imprime a raiz a quarta de um numero";
 imprima "\n F - Finalizar";
 imprima "\nOPCAO: ";
 leia op;
 se(op  ==  "A" || op  ==  "a" )
 {
  imprima "\ndigite 1 palavra: ";
  leia p1;
  imprima "\ndigite 2 palavra: ";
```

```
 leia p2;
 imprima "\ndigite 3 palavra: ";
 leia p3;
 se(strtam(p1)< strtam(p2))
 { menor <- p1; }
 senao
 {menor <- p2;}
 se(strtam(p3)< strtam(menor))
 { menor <- p3; }
  imprima "\n",menor;
}
senao
{
 se(op  ==  "B" || op  ==  "b" )
 {imprima "\ndigite uma palavra: ";
  leia p1;
  se(strtam(p1) >5)
  { n <- strtam(p1)-2;
    p2<-strprim(p1);
    p3<-strnresto(p1,n);
    np<-strconcat(p2,p3);
    imprima "\n",np;
   }
   senao
   {
     n<- strtam(p1) - 3;
     np<-strnresto(p1,n);
     imprima "\n",np;
    }
  }
  senao
  {
   se(op  ==  "C" || op  ==  "c")
   {
       imprima "\ndigite um numero: ";
       leia num;
       se(num >= 0.)
       {
         imprima "\nraiz a quarta: ",formatar(num ** (1/4),2);
       }
       senao
       {   imprima "\nNAO PODE SER FEITA";     }
    }
    senao
    {
      se(op  ==  "F" || op  ==  "f")
      {  imprima "\n\n\n\t\t\t\tFINALIZANDO";  }
      senao
```

```
        {  imprima "\nopcao invalida";  }
      }
    }
   }
   imprima "\n\n\n";
  }
  enquanto( op  < >  "F" && op  < >  "f")
  imprima "\n";
fimprog
```

VÍDEO
MENU - um pouco de tudo *A - Armazena na variavel menor e imprime o nome que tiver o menor numero de caracteres* *B - Gera e imprime uma nova palavra* *C - Calcula e imprime a raiz a quarta de um numero* *F - Finalizar* *OPCAO:c* *digite um numero: 16.* *raiz a quarta:2.00*   *MENU - um pouco de tudo* *A - Armazena na variavel menor e imprime o nome que tiver o menor numero de caractesres* *B - Gera e imprime uma nova palavra* *C - Calcula e imprime a raiz a quarta de um numero* *F - Finalizar* *OPCAO:f*   *FINALIZANDO*

DICAS:

*1. **Não** use **enquanto** no lugar do **para**, só se não tiver alternativa.*

*2. Na dúvida entre **enquanto** e **faca .. enquanto**, use **enquanto**.*

*3. Para simular a estrutura do **faca ... enquanto**, usando a estrutura do **enquanto**, basta inicializar a variável da condição com qualquer valor válido.*

Simulando a estrutura do faca ... enquanto

```
prog menu
 string  op,menor,p1,p2,p3,np;
 int n;
 real num;
 op <- "";
 enquanto(op< >"F" && op< >"f")
 {
 imprima "\n MENU - um pouco de tudo";
 imprima "\n A - Armazena na variavel menor e imprime o nome que tiver o
 menor numero de caracteres" ;
 imprima "\n B - Gera e imprime uma nova palavra";
 imprima "\n C - Calcula e imprime a raiz a quarta de um numero";
 imprima "\n F - Finalizar";
 imprima "\nOPCAO: ";
 leia op;
 se(op  ==  "A" || op  ==  "a" )
 { ...
     senao
     {  imprima "\nopcao invalida";  }
   }}}
   imprima "\n\n\n";
 }
fimprog
```

> *MENU - um pouco de tudo*
> *A - Armazena na variavel menor e imprime o nome que tiver o menor numero de caracteres*
> *B - Gera e imprime uma nova palavra*
> *C - Calcula e imprime a raiz a quarta de um numero*
> *F - Finalizar*
> *OPCAO:f*
>
> *FINALIZANDO*

EXERCÍCIOS – LISTA 4

ENQUANTO E FACA ENQUANTO

algoritmo 262

Entrar com números e imprimir o triplo de cada número. O algoritmo acaba quando entrar o número –999.

```
prog enq1
 real num;
 imprima "\ndigite numero ou -999. para terminar: ";
 leia num;
 enquanto(  num < > -999. )
 {
   imprima  "\ntriplo: ", num * 3;
   imprima "\ndigite numero ou -999. para  terminar:  ";
   leia num;
 }
 imprima "\n";
fimprog
```

algoritmo 263

Entrar com números enquanto forem positivos e imprimir quantos números foram digitados.

```
prog enq2
 int a;
 real num;
 a <- 0;
 imprima "\ndigite numero positivo: ";
 leia num;
 enquanto(  num > 0. )
 {
  a++;
   imprima "\nigite numero positivo: ";
   leia num;
 }
 imprima "\ntotal: ", a;
 imprima "\n";
fimprog
```

algoritmo 264

Entrar com vários números positivos e imprimir a média dos números digitados.

```
prog enq3
 int a;
 real num, soma;
 a <- 0;
 soma <-  0.;
 imprima "\ndigite numero positivo: ";
 leia num;
 enquanto(  num > 0.)
 {
   a++;
   soma  <-  soma + num;
```

```
    imprima "\ndigite numero positivo: ";
    leia num;
  }
  imprima "\nmédia: ", soma / a;
  imprima "\n";
fimprog
```

algoritmo 265

Ler vários números e informar quantos números entre 100 e 200 foram digitados. Quando o valor 0 (zero) for lido, o algoritmo deverá cessar sua execução.

```
prog enq4
 int a;
 real num;
 a   <-  0;
 imprima "\ndigite numero ou 0 para sair: ";
 leia num;
 enquanto(  num < > 0.)
 {
    se(num>=100. && num <=200.)
    {   a  ++;     }
   imprima "digite numero ou 0 para sair: ";
   leia num;
 }
 imprima "\ntotal: ",  a;
 imprima "\n";
fimprog
```

algoritmo 266

Entrar com nomes enquanto forem diferentes de FIM e imprimir o primeiro caractere de cada nome.

```
prog enq5
string nome;
 imprima "\ndigite nome ou FIM para terminar:  ";
 leia nome;
 enquanto(  nome < >  "FIM" && nome < >"fim" && nome < >"Fim"  )
 {
  imprima "\n",strprim(nome);
  imprima "\ndigite nome ou FIM para terminar:  ";
  leia nome;
 }
 imprima "\n";
fimprog
```

algoritmo 267

Entrar com profissão de várias pessoas e imprimir quantos são dentistas (considerar DENTISTA, dentista e Dentista).

```
prog enq6
 int a, c;
 string prof;
 a   <-  0;
 imprima "\ndigite profissão ou FIM para sair: ";
 leia prof;
 enquanto(  prof  < >  "FIM" && prof < >"fim" && prof < >"Fim")
 {
  se( prof ==  "DENTISTA" || prof == "Dentista" || prof =="dentista" )
  { a++; }
  imprima "\ndigite profissão ou FIM para sair: ";
  leia prof ;
 }
 imprima "\ntotal de dentistas: ", a;
 imprima "\n";
fimprog
```

algoritmo 268

Entrar com sexo de várias pessoas e imprimir quantas pessoas são do sexo masculino (considerar m ou M).

```
prog enq7
 int a;
 string sexo;
 a   <-  0;
 imprima "\ndigite sexo(M/F) ou @ para sair: ";
 leia sexo;
 sexo <- strprim(sexo);
 enquanto(  sexo < >  "@" )
 {
  se(sexo == "M" || sexo == "m" )
  { a++; }
  imprima "\ndigite sexo(M/F) ou @ para sair: ";
  leia sexo;
  sexo <- strprim(sexo);
 }
 imprima "\ntotal de pessoas do sexo masculino: ", a;
 imprima "\n";
fimprog
```

algoritmo 269

Entrar com números e imprimir o quadrado de cada número até entrar um número múltiplo de 6 que deverá ter seu quadrado também impresso.

```
prog enq8
 int num;
 faca
 {
   imprima "\ndigite numero ou multiplo de 6 para acabar: ";
   leia num;
   imprima  "quadrado: ", num ^2;
 }
 enquanto( num % 6 < > 0 )
 imprima "\n";
fimprog
```

algoritmo 270

Ler vários números até entrar o número -999. Para cada número, imprimir seus divisores.

```
prog enq9
 int num, a;
 imprima "\ndigite numero ou -999 para acabar: ";
 leia num;
 enquanto(num< >-999)
 {
  imprima "\n1  ";
  para (a <- 2; a <= num div 2; a++ )
  {
   se( num  % a == 0 )
   {  imprima a, "  "; }
  }
 imprima num;
 imprima "\ndigite numero ou -999 para acabar: ";
 leia num;
 }
 imprima "\n";
fimprog
```

algoritmo 271

Dado um país A, com 5.000.000 de habitantes e uma taxa de natalidade de 3% ao ano, e um país B com 7.000.000 de habitantes e uma taxa de natalidade de 2% ao ano, calcular e imprimir o tempo necessário para que a população do país A ultrapasse a população do país B.

```
prog enq10
 int anos;
 real  a, b;
 a  <-  5000000.;
 b  <-  7000000.;
 anos  <-  0;
 enquanto( a <= b )
 {
   a  <-   a * 1.03;
   b  <-    b * 1.02;
   anos++;
 }
 imprima  "\nANOS: ",anos;
 imprima "\n";
fimprog
```

algoritmo 272

Chico tem 1,50m e cresce 2 centímetros por ano, enquanto Juca tem 1,10m e cresce 3 centímetros por ano. Construir um algoritmo que calcule e imprima quantos anos serão necessários para que Juca seja maior que Chico.

```
prog enq11
 int anos;
 real c, j;
 c <- 1.5;
 j <- 1.1;
 anos <-  0;
 enquanto(  j <= c  )
 {
    c <- c + 0.02;
    j <- j + 0.03;
    anos++;
 }
 imprima "\nanos: ",anos;
 imprima "\n";
fimprog
```

algoritmo 273

Uma empresa de fornecimento de energia elétrica faz a leitura mensal dos medidores de consumo. Para cada consumidor, são digitados os seguintes dados:

- *número do consumidor*
- *quantidade de kWh consumidos durante o mês*
- *tipo (código) do consumidor*

  1 – residencial, preço em reais por kWh = 0,3
  2 – comercial, preço em reais por kWh = 0,5
  3 – industrial, preço em reais por kWh = 0,7

Os dados devem ser lidos até que seja encontrado um consumidor com número 0 (zero). Calcular e imprimir:

- *o custo total para cada consumidor*
- *o total de consumo para os três tipos de consumidor*
- *a média de consumo dos tipos 1 e 2*

```
prog enq12
 int c12, codigo;
 real  qtde, total, total12;
 c12  <-  0;
 total  <-  0.;
 total12  <-  0.;
 imprima "\ndigite codigo(1-res 2-com 3-ind) ou 0 para sair: ";
 leia codigo;
 enquanto( codigo < > 0)
 {
  imprima "digite quantidade em KWH: ";
  leia qtde;
  total  <-  total + qtde;
  se(codigo ==1)
  {
    imprima  "\ncusto: ",qtde * 0.3;
    c12  <-   c12 + 1;
    total12  <-  total12 + qtde;
  }
  senao
  {
   se(codigo == 2)
   {
     imprima  "\ncusto: ",qtde * 0.5;
     c12  <- c12 + 1;
     total12  <- total12 + qtde;
    }
    senao
    {
      se(codigo == 3)
      {
       imprima  "\ncusto: ",qtde * 0.70;
      }
    }
 }
 imprima "\ndigite codigo(1-res 2-com 3-ind) ou 0 para sair: ";
 leia codigo;
 }
 imprima  "\ntotal consumido em 1-2-3: ", total ;
 se(c12< >0)
 {
```

```
 imprima  "\nmédia 1-2: ", total12 / c12 ;
 }
 senao
 { imprima "\nNenhum consumidor na faixa 1-2";}
 imprima "\n";
fimprog
```

algoritmo 274

Criar um algoritmo que deixe entrar com 10 números positivos e imprima raiz quadrada de cada número. Para cada entrada de dados deverá haver um trecho de proteção para que um número negativo não seja aceito.

```
prog enq13
 real num;
 int a;
 para(a <- 1;a<=10;a++)
 {
   imprima "\nEntre com ",a, "º numero: ";
   leia num;
   enquanto(num <= 0.)
   {   imprima "\nATENCAO! Entre com ",a, "º numero maior do que 0: ";
       leia num;
    }
    imprima "\nRaiz quadrada: ", raiz(num), "\n";
 }
 imprima "\n";
fimprog
```

algoritmo 275

Criar um algoritmo que leia vários números inteiros e apresente o fatorial de cada número. O algoritmo se encerra quando se digita um número menor do que 1.

```
prog enq14
 int num, fat, c;
 imprima "\nDigite numero inteiro positivo para calculo do fatorial ou
 menor do que 1 para terminar: ";
 leia num;
 enquanto(num > 0)
 { fat <- 1;
  para(c <-2; c <= num; c++)
  { fat <- fat * c;}
  imprima "\nfatorial: ",fat;
  imprima "\nDigite numero inteiro positivo para calculo do fatorial ou
           menor do que 1 para terminar: "; # continuacao
  leia num;
 }
 imprima "\n";
fimprog
```

algoritmo 276

Entrar com a idade de várias pessoas e imprimir:

- *total de pessoas com menos de 21 anos*
- *total de pessoas com mais de 50 anos.*

```
prog enq15
 int idade, c21, c50;
 c21  <-  0;
 c50  <-  0;
 imprima "\nEntre com uma idade: ";
 leia idade;
 enquanto(idade >=0 && idade <= 140)
 {
   se(idade < 21)
   { c21++; }
   senao
   {
     se(idade > 50)
     { c50++; }
   }
  imprima "\nEntre com uma idade: ";
  leia idade;
  }
 imprima "\nTotal de idades inferiores a 21: ", c21;
 imprima "\nTotal de idades maiores do que 50: ", c50;
 imprima "\n";
fimprog
```

algoritmo 277

Entrar com vários números e verificar se eles são ou não quadrados perfeitos. O algoritmo termina quando se digita um número menor ou igual a 0.

↳ *Um número é quadrado perfeito quando tem um número inteiro como raiz quadrada.*

```
prog enq16
 int num,c,p;
 imprima "\nDigite numero maior do que 0: ";
 leia num;
 enquanto(num > 0)
 {
  c <- 1;
  p <- c * c;
 enquanto( p < num)
```

```
 { c++;
   p <- c * c;
 }
 se(p == num)
 { imprima "\nQuadrado perfeito";}
 senao
 { imprima "\nNao e quadrado perfeito";}
 imprima "\nDigite numero maior do que 0: ";
 leia num;
 }
 imprima "\n";
fimprog
```

algoritmo 278

Entrar com um número e verificar se ele é um número primo.

```
prog enq17
 int num,cont,num1;
 imprima "\nEntre com o numero: ";
 leia num;
 cont  <-  0;
 num1  <-  2;
 enquanto(cont == 0 && num1<= num div  2)
 {
    se( num % num1==0 )
    { cont <- 1;  }
    num1++;
 }
 se( cont == 1 )
 { imprima "\nnao e primo "; }
 senao
 { imprima "\ne primo" ;}
 imprima"\n";
fimprog
```

algoritmo 279

Escrever um algoritmo que receba vários números inteiros e imprima a quantidade de números primos dentre os números que foram digitados. O algoritmo acaba quando se digita um número menor ou igual a 0.

```
prog enq18
 int num, cont, num1, primo;
 imprima "\nEntre com o numero maior do que 0: ";
 leia num;
 primo <-  0;
 enquanto(num > 0)
 {
```

```
 cont  <-  0;
 num1  <-  2;
 enquanto(cont == 0 && num1<= num div   2)
 {
   se( num % num1== 0 )
   { cont <- 1;  }
     num1++;
 }
 se( cont < > 1 )
 { primo++;  }
 imprima "\nEntre com o numero maior do que 0: ";
 leia num;
 }
 imprima "\nTotal de primos: ", primo;
 imprima "\n";
fimprog
```

algoritmo 280

Entrar com um número e verificar se ele é um número triangular.

Um número é triangular quando é resultado do produto de três números consecutivos. Exemplo: 24 = 2 x 3 x 4

```
prog enq19
 int num, p, num1;
 num1 <- 1;
 imprima  "\ndigite numero: ";
 leia num;
 p  <- num1 * (num1+1) * (num1+2);
 enquanto(p < num )
 { num1++;
  p  <- num1 *(num1+1)*(num1+2);
  }
 se( p==num )
 { imprima  "\nE triangular ";}
 senao
 { imprima  "\nNao e triangular ";}
 imprima "\n";
fimprog
```

algoritmo 281

Entrar com vários números e imprimir o maior número. O algoritmo acaba quando se digita -9999.

```
prog enq20
```

```
 real num, maior;
 imprima "\ndigite numero ou - 999. para acabar: " ;
 leia num;
 maior  <-  num;
 enquanto( num < > -999.)
 {
   se( num > maior)
   { maior  <-  num ;}
   imprima "\ndigite numero ou - 999. para acabar: " ;
   leia num;
 }
 imprima "\nmaior: ", maior ;
 imprima "\n";
fimprog
```

algoritmo 282

Entrar com o número da conta e o saldo de várias pessoas. Imprimir todas as contas, os respectivos saldos e uma das mensagens: positivo/negativo. Ao final, o percentual de pessoas com saldo negativo. O algoritmo acaba quando se digita um número negativo para a conta.

```
prog enq21
 int conta, cneg, c;
 real  saldo;
 c  <-  0;
 cneg  <-  0;
 imprima "\ndigite conta ou numero negativo para terminar: ";
 leia conta;
 enquanto(  conta > 0)
 {
   c++;
   imprima "\nsaldo:";
   leia saldo;
   se( saldo < 0. )
   {   cneg++;
      imprima conta, "-", saldo, "- negativo " ;
   }
   senao
   {   imprima conta, " - ",saldo, "- positivo " ;   }
   imprima "\ndigite conta ou numero negativo para terminar: ";
   leia conta;
 }
 imprima  "\npercentual negativo: ", cneg * 100 / c,  "% " ;
 imprima "\n";
fimprog
```

algoritmo 283

Uma agência de uma cidade do interior tem, no máximo, 10.000 clientes. Criar um algoritmo que possa entrar com número da conta, nome e saldo de cada cliente. Imprimir todas as contas, os respectivos saldos e uma das mensagens: positivo / negativo. A digitação acaba quando se digita -999 para número da conta ou quando chegar a 10.000. Ao final, deverá sair o total de clientes com saldo negativo, o total de clientes da agência e o saldo da agência.

```
prog enq22
 int conta, cneg, c;
 real  saldo, soma;
 string nome;
 c  <-  0;
 cneg  <-  0;
 soma  <-  0.;
 imprima "\ndigite conta ou -999 para terminar: ";
 leia conta;
 enquanto(  conta > 0 &&  c < 10000)
 {
  c ++;
  imprima "\nnome: ";
  leia nome;
  imprima "\nsaldo: ";
  leia saldo;
  soma  <-  soma + saldo;
  se( saldo < 0. )
  {  cneg     <-  cneg +1;
     imprima conta, "-", saldo,"- negativo " ;
  }
  senao
  {  imprima conta, " - ",saldo, "- positivo " ;  }
  imprima "\ndigite conta ou -999 para terminar ";
   leia conta;
 }
 imprima  "\nTotal de clientes com saldo negativo: ", cneg ;
 imprima  "\nTotal de clientes da agencia: ", c ;
 imprima "\n";
fimprog
```

algoritmo 284

Criar um algoritmo que leia uma seqüência de números terminada por 0 e imprima o número que for múltiplo de sua posição na seqüência. Exemplo:

valores lidos:	*3*	*7*	*8*	*16*
posição:	*1*	*2*	*3*	*4*
impressão:	*3*	*16*		

```
prog enq23
 int num, pos;
 imprima "\nDigite um número ou zero para acabar: ";
 leia num;
 pos  <-  0;
 enquanto(num < > 0)
 {
   pos++;
   se(num % pos == 0)
   {
     imprima "\n",num;
   }
   imprima "\nDigite um número ou zero para acabar: ";
   leia num;
 }
 imprima "\n";
fimprog
```

algoritmo 285

Sabendo-se que a UAL calcula a divisão através de subtrações sucessivas, criar um algoritmo que calcule/imprima o resto da divisão de números inteiros lidos. Suponha que os números lidos sejam positivos e que o dividendo seja maior do que o divisor.

```
prog enq24
 int dividendo, divisor, aux, resto;
 imprima "\ndigite dividendo: " ;
 leia dividendo;
 imprima "\ndigite divisor: " ;
 leia divisor;
 se(dividendo < divisor)
 {
 aux  <- dividendo;
 dividendo <-  divisor;
 divisor <-  aux;
 }
 resto  <- dividendo - divisor;
 enquanto( resto > divisor)
 {  dividendo <- resto;
   resto  <- dividendo - divisor;
 }
 imprima "\nResto: ", resto;
 imprima "\n";
fimprog
```

algoritmo 286

Criar um algoritmo que calcule o M.M.C. entre dois números lidos.

```
prog enq25
 int a, b, aux, soma;
 imprima "\nEntre com 1 numero: ";
 leia a;
 imprima "\nEntre com 2 numero: ";
 leia b;
 se(a < b)
 {  aux  <- a;  a  <- b;  b  <-  aux;}
 soma  <- a;
 enquanto(soma % b < > 0)
 {
  soma  <- soma + a;
 }
 imprima "\nM.M.C: ",soma;
 imprima "\n";
fimprog
```

algoritmo 287

Criar um algoritmo que calcule o M.D.C. entre dois números inteiros lidos.

```
prog enq26
 int a, b, aux, resto;
 imprima "\nEntre com 1 numero: ";
 leia a;
 imprima "\nEntre com 2 numero: ";
 leia b;
 se(a < b)
 {  aux  <- a;  a  <- b;  b  <-  aux;}
 resto  <- a % b;
 enquanto(resto < >0)
 {   a  <- b;   b  <-  resto;   resto  <- a % b;}
 imprima "\nM.D.C: ",b;
 imprima "\n";
fimprog
```

algoritmo 288

Repare a seguinte característica do número 3025: 30 + 25 = 55 e 55^2 = 3025

Criar um algoritmo que possa ler vários números inteiros de 4 algarismos, um de cada vez, e diga se o número apresenta a mesma característica (repare que 3025 / 100 = 30 com resto 25). O algoritmo termina quando for lido um valor menor que 1.000 ou maior que 9999.

```
prog enq27
 int num, dc, du, q;
```

```
 imprima "\nEntre com um numero de 4 algarismos: ";
 leia num;
 enquanto(num >= 1000 && num <=9999)
 {
  dc  <- num div   100;
  du  <- num % 100;
  q <-  (dc + du)^ 2;
  se(q == num)
  {imprima "\nPossui a caracteristica";}
  senao
   {imprima "\nNao possui a caracteristica";}
 imprima "\nEntre com um numero de 4 algarismos: ";
 leia num;
 }
 imprima "\n";
fimprog
```

algoritmo 289

Criar um algoritmo que entre com vários números inteiros e positivos e imprima a média dos números múltiplos de 3.

```
prog  enq28
 int num, cont, soma;
 soma <- 0; cont <- 0;
 imprima "\ndigite numero positivo e para terminar, negativo ou 0: ";
 leia num;
 enquanto(num > 0)
 { se(num % 3 == 0 )
   {cont++;   soma <- soma + num; }
   imprima "\ndigite numero positivo e para terminar, negativo ou 0: ";
   leia num; }
 se( cont <> 0)
 { imprima  "\nmedia dos numeros multiplos de 3: ", soma / cont; }
 senao { imprima "\nnenhum numero multiplo de 3: ";}
 imprima "\n";
fimprog
```

algoritmo 290

Criar um algoritmo que entre com vários números inteiros positivos e imprima o produto dos números ímpares digitados e a soma dos pares.

```
prog  enq29
 int num, prod, soma;
 soma <- 0;
 prod <- 1;
 imprima "\ndigite numero positivo, zero ou negativo para terminar: ";
 leia num;
```

```
 enquanto(num > 0)
 {  se(num % 2 == 0 )
   { soma <- soma + num; }
   senao
   { prod <- prod * num; }
   imprima "\ndigite numero positivo, zero  ou negativo para terminar: ";
 leia num;
 }
 imprima  "\nproduto dos numeros impares: ", prod;
 imprima  "\nsoma dos numeros pares: ", soma;
 imprima "\n";
fimprog
```

algoritmo 291

Criar um algoritmo que possa ler um conjunto de pedidos de compra e calcule o valor total da compra. Cada pedido é composto pelos seguintes campos:

- *número de pedido*
- *data do pedido (dia, mês, ano)*
- *preço unitário*
- *quantidade*

O algoritmo deverá processar novos pedidos ate que o usuário digite (zero) como número do pedido.

```
prog enq30
 int data, np,quant;
 real   pu, val,soma;
 imprima "\nDigite número do pedido ou 0 para terminar: ";
 leia  np;
 soma  <-  0.;
 enquanto( np > 0 )
 {
  imprima "\nDigite data: ";
  leia  data;
  imprima "\nDigite preço unitário: ";
  leia  pu;
  imprima "\nDigite número quantidade: ";
  leia  quant;
   val  <-   pu * quant;
   soma  <-  soma + val;
   imprima "\no valor da compra  na data ", data, " é  ", val ;
   imprima "\nDigite número do pedido ou 0 para terminar: ";
   leia  np;
 }
 imprima "o valor total da compra e:  ",soma;
 imprima "\n";
fimprog
```

algoritmo 292

Criar um algoritmo que leia vários números terminados por 0 e imprima o maior, o menor e a média aritmética dos números. O número 0 (zero) não faz parte da seqüência.

```
prog enq31
 real num, maior, menor, soma;
 int a;
 imprima "\nEntre com um numero ou 0 para terminar: ";
 leia num;
 a  <- 0;
 soma  <- 0.;
 maior  <- num;
 menor  <- num;
 enquanto(num < > 0.)
 {  a++;
   soma  <-  soma + num;
   se(num > maior)
   {maior  <-  num;}
   senao
   { se(num<menor)
      { menor  <-  num; }
   }
 imprima "\nEntre com um numero ou 0 para terminar: ";
 leia num;
 }
 imprima "\nMaior: ", maior;
 imprima "\nMenor: ", menor;
 imprima "\nMedia: ", soma/a;
 imprima "\n";
fimprog
```

algoritmo 293

Criar um algoritmo que leia idade e sexo(0-masculino, 1-feminino) de várias pessoas. Calcule e imprima a idade média, total de pessoas do sexo feminino com idade entre 30-45 inclusive e o número total de pessoas do sexo masculino. O algoritmo termina quando se digita 0 para idade.

```
prog enq32
 real media;
 int totpessoas, masc, fem,sexo, soma, idade;
 soma  <-  0;
 masc  <-  0;
 fem  <-   0;
 totpessoas <- 0;
 imprima "\nEntre a idade ou 0 para terminar: ";
 leia idade;
 enquanto(idade > 0)
```

```
{
 totpessoas  ++;
 soma  <-  soma + idade;
 imprima "\nEntre sexo(0-masculino e 1- feminino): ";
 leia sexo;
 se(sexo == 0)
 { masc ++; }
 senao
 {
   se(idade >= 30 && idade <= 45)
     {fem ++;}
 }
 imprima "\nEntre a idade ou 0 para terminar: ";
 leia idade;
}
media  <-  soma / totpessoas;
imprima "\nMasculino: ", masc;
imprima "\nFeminino de 30 ate 45: ", fem;
imprima "\nMedia: ", media;
imprima "\n";
fimprog
```

algoritmo 294

Uma das maneiras de se conseguir a raiz quadrada de um número é subtrair do número os ímpares consecutivos a partir de 1, até que o resultado da subtração seja menor ou igual a zero. 0 número de vezes que se conseguir fazer a subtração é a raiz quadrada exata (resultado 0) ou aproximada do número (resultado negativo).

Exemplo: *Raiz de 16*

$$16 - 1 = 15 - 3 = 12 - 5 = 7 - 7 = 0$$

A raiz de 16 é 4.

```
prog  enq33
 int i, num, cont;
 cont <- 0;
 i <- 1;
 imprima "\ndigite numero positivo ou negativo para terminar: ";
 leia num;
 enquanto(num > 0)
 {  num <- num - i;   i <- i + 2;   cont++;}
 se( num == 0)
 { imprima "\na raiz quadrada e: ", cont;}
 senao
 { imprima "\na raiz quadrada aproximada esta entre: ", cont -1, " e ", cont;}
 imprima "\n";
fimprog
```

algoritmo 295

Uma fábrica produz e vende vários produtos e para cada um deles tem-se o nome, quantidade produzida e quantidade vendida. Criar um algoritmo que imprima:

- *Para cada produto, nome, quantidade no estoque e uma mensagem se o produto tiver menos de 50 itens no estoque;*
- *Nome e quantidade do produto com maior estoque;*

```
prog enq34
 int qte, qtp, np, mqte, a, qtv;
 string nome, mnome;
 a <-1;
 mqte <-0;
 imprima "\nDigite a quantidade de produtos da fabrica : ";
 leia np;
 enquanto(a <= np)
 {
   imprima "\n\nDigite o nome do produto ",a,": ";
   leia nome;
   imprima "\nDigite a quantidade produzida :";
   leia qtp;
   imprima "\nDigite a quantidade vendida :";
   leia qtv;
   qte <- qtp - qtv;
   se(qte > mqte)
    {
      mqte <-qte;
      mnome <-nome;
     }
   imprima "\nProduto :", nome;
   imprima "\nA quantidade deste produto  de : ", qte, " itens";
   se(qte<50)
   { imprima "\nIMPORTANTE! Produto precisa de uma maior producao!";  }
   a++;
 }
 imprima "\nO produto com maior estoque : ", mnome, "\ncom : ", mqte, " pecas";
 imprima "\n";
fimprog
```

algoritmo 296

Na Usina de Angra dos Reis, os técnicos analisam a perda de massa de um material radioativo. Sabendo-se que este perde 25% de sua massa a cada 30 segundos, criar um algoritmo que imprima o tempo necessário para a que massa deste material se torne menor que 0,10 grama. O algoritmo pode calcular o tempo para várias massas.

```
prog enq35
 int contempo;
 real massa, tempo;
 string resp;
 contempo <- 0;
 imprima "\nDigite S se desejar novo calculo ou qualquer letra para terminar: ";
 leia resp;
 enquanto(resp == "S"|| resp == "s")
 {
  imprima "\nDigite a massa em gramas do material :";
  leia massa;
  enquanto(massa >= 0.10)
  {
    contempo++;
    massa <- 0.75 * massa;
  }
  tempo <- (contempo*30) / 60;
  imprima "\nO tempo foi de :", tempo, "minutos.";
  imprima "\n";
  imprima "\nDigite S se desejar novo calculo ou qualquer letra para terminar:
  ";
  leia resp;
 }
 imprima "\n";
fimprog
```

algoritmo 297

Uma pousada estipulou o preço para a sua diária em R$ 30,00 e mais uma taxa de serviços diários de:

- *R$ 15,00, se o número de dias for menor que 10;*
- *R$ 8,00, se o número de dias for maior ou igual a 10;*

Criar um algoritmo que imprima nome, conta e o número da conta de cada cliente e ao final o total ganho pela pousada.

```
prog enq36
 int contac, ndias, tot, totc;
 string nome;
 tot <-0;
 totc <-0;
 imprima "Digite o numero da conta do cliente ou zero para sair: ";
 leia contac;
 enquanto(contac< >0)
 {
     imprima "\nDigite o nome do cliente:";
     leia nome;
```

```
    imprima "\nQuantos dias vai ser a estadia dele ?";
    leia ndias;
    se(ndias < 10)
    { totc <-(ndias*30) + (ndias*15); }
    senao
    { totc <-(ndias*30) + (ndias*8);  }
    tot <-tot +totc;
    imprima "\nCliente :", nome, "\nConta numero :", contac, "\nSaldo
    final : R$ ",totc;
    imprima "\n\nDigite o numero da conta do cliente ou zero para sair: ";
    leia contac;
 }
 imprima "\nO total arrecadado pela pousada foi de : R$ ", tot;
 imprima "\n";
fimprog
```

algoritmo 298

Numa universidade, os alunos das turmas de informática fizeram uma prova de algoritmos. Cada turma possui um número de alunos. Criar um algoritmo que imprima:

- *quantidade de alunos aprovados;*
- *média de cada turma;*
- *percentual de reprovados.*

↳ *Considere aprovado com nota >=7.0.*

```
prog enq37
 int nt, a, x, na, contap, total;
 real nota, sn;
 a <- 1;
 contap <- 0;
 total <- 0;
 imprima "Digite o numero de turmas :";
 leia nt;
 enquanto(a<=nt)
 {
   sn <-0.;
   imprima "\n\nDigite o numero de alunos da turma ", a, ": ";
   leia na;
   total <- total + na;
   x <- 1;
   enquanto(x <= na)
   {
```

```
        imprima "\nDigite a nota do aluno ", x, ": ";
        leia nota;
        se(nota >=7.0)
        { contap++;   }
        sn <- sn + nota;
        x++;
     }
     imprima "\nMedia da turma ", a, ": ", sn/na;
     a++;
  }
  imprima "\n\nForam aprovados ", contap, " alunos.";
  imprima "\n\nPercentual de alunos reprovados  ", formatar(
            (total-contap)* 100/total,2), "%"; # continuacao
  imprima "\n";
fimprog
```

algoritmo 299

Os alunos de informática tiveram cinco provas: 1, 2, 3, 4 e 5. Criar um algoritmo que imprima:

- *Nome dos que foram aprovados em todas as matérias;*
- *Nome dos alunos aprovados nas matérias 1 e 4;*
- *A porcentagem dos aprovados na matéria 3.*

Considere aprovado com nota >=7.0.

```
prog enq38
 int x, qa, cont;
 real n1, n2, n3, n4, n5;
 string nome;
 x <-0;
 imprima "\nDigite a quantidade de alunos :";
 leia qa;
 cont <- 0;
 enquanto(x < qa)
 {
    imprima "\nDigite nome:";
    leia nome;
    imprima "\nDigite a nota na prova 1 :";
    leia n1;
    imprima "\nDigite a nota na prova 2 :";
    leia n2;
    imprima "\nDigite a nota na prova 3 :";
    leia n3;
    imprima "\nDigite a nota na prova 4 :";
```

```
    leia n4;
    imprima "\nDigite a nota na prova 5 :";
    leia n5;
    se(n1 >= 7.0 && n4 >= 7.0)
    {
      se(n2 >= 7.0 && n3 >= 7.0 && n5 >= 7.0)
      {
        imprima "\nParabens! ", nome, " voce foi aprovado em todas as materias!";
      }
      senao
      {imprima "\n", nome, " voce passou nas materias 1 e 4"; }
    }
    se(n3 >= 7.0)
    { cont++;    }
    x++;
 }
 imprima "\n\nA porcentagem dos aprovados na materia 3  :",formatar(
          (cont*100)/qa,2); # continuacao
 imprima "\n";
fimprog
```

algoritmo 300

Uma pesquisa de opinião realizada no Rio de Janeiro, teve as seguintes perguntas:

Qual seu time de coração?
1-Fluminense;
2-Botafogo;
3-Vasco;
4-Flamengo;
5-Outros
Onde você mora?
1-RJ;
2-Niterói;
3-Outros;
Qual o seu salário?
Criar um algoritmo que imprima:

- *o número de torcedores por clube;*
- *a média salarial dos torcedores do Botafogo;*
- *o número de pessoas moradoras do Rio de Janeiro, torcedores de outros clubes;*
- *o número de pessoas de Niterói torcedoras do Fluminense;*

↳ *O algoritmo acaba quando se digita 0 para o time.*

```
prog enq39
 int np, x, op, op2, contflu, contbot, contvas, contfla, contout,
 int contflu2, contout2;
 real sal, msal;
 x <- 0;
 contflu <- 0;
 contflu2 <- 0;
 contbot <- 0;
 msal <-0.;
 contvas <- 0;
 contfla <- 0;
 contout <- 0;
 contout2 <- 0;
 imprima "\nDigite 1 FLU  2 BOTA 3 VASCO 4 FLA 5 Outros 0 sair: ";
 leia op;
 enquanto(op <> 0)
 { enquanto(op < 1 || op > 5)
   { imprima "\nDigite 1 FLU  2 BOTA 3 VASCO 4 FLA 5 Outros: ";
     leia op;
  }
  x++;
  imprima "\nOnde voce mora 1 RJ 2 Niteroi 3 Outros ?";
  leia op2;
  imprima "\nQual o seu salario ?"; leia sal;
  se( op == 1 )
  { contflu++;
    se( op2==2 )
    { contflu2++; }
  }
  senao
  { se( op == 2 )
    {contbot++; msal <-msal + sal; }
     senao
     { se( op == 3 )
       {contvas++; }
       senao
       { se( op == 4 )
         { contfla++;}
         senao
         { se( op == 5 )
           { contout++;
             se( op2==1 )
             { contout2++; }
           } } } } }
  imprima "\nDigite 1 FLU  2 BOTA 3 VASCO 4 FLA 5 Outros 0 sair: ";
  leia op;
 }
```

```
imprima "\nTotal de torcedores do Fluminense :", contflu;
imprima "\nTotal de torcedores do Botafogo :", contbot;
imprima "\nTotal de torcedores do Vasco :", contvas;
imprima "\nTotal de torcedores do Flamengo :", contfla;
imprima "\nTotal de torcedores de outros clubes :", contout;
se( contbot <> 0 )
{ imprima "\nMedia salarial dos torcedores do Botafogo :",
            msal/contbot; } # continuacao
senao
{ imprima "\nNenhum torcedor do Botafogo";}
imprima "\nTotal de pessoas do Rio de Janeiro torcedores de outros
          clubes :",contout2; # continuacao
imprima "\nTotal de pessoas de Niteroi torcedores do Fluminense :",
          contflu2; # continuacao
imprima "\n";
fimprog
```

algoritmo 301

Numa universidade cada aluno possui os seguintes dados:

- *Renda pessoal;*
- *Renda familiar;*
- *Total gasto com alimentação;*
- *Total gasto com outras despesas;*

Criar um algoritmo que imprima a porcentagem dos alunos que gasta acima de R$ 200,00 com outras despesas, o número de alunos com renda pessoal maior que renda familiar e a porcentagem gasta com alimentação e outras despesas em relação às rendas pessoal e familiar.

O algoritmo acaba quando se digita 0. para renda pessoal.

```
prog enq40
 int n,  cont, cont2;
 real rp, rf, totali, totout;
 n <-0;
 cont <- 0;
 cont2 <- 0;
 imprima "\nDigite sua renda pessoal ou 0. para acabar: ";
 leia rp;
 enquanto(rp > 0.)
 { n++;
  imprima "\nDigite renda familiar fora a sua: ";
  leia rf;
  imprima "\nDigite quanto gasta com alimentacao: ";
```

```
  leia totali;
  imprima "\nDigite quanto gasta com outras despesas: ";
  leia totout;
  se(totout >200.)
  {  cont++;   }
  se(rp > rf)
  {  cont2++;   }
  imprima "\n", ((totali+totout)*100)/(rp+rf), "% sao gastos com
  alimentos + despesas em relacao a renda pessoal e familiar";
  imprima "\n\nDigite sua renda pessoal ou 0. para acabar: ";
  leia rp;
 }
 imprima"\n", formatar((cont*100)/n,2), "% dos alunos gastam acima de
 R$ 200,00 com outras despesas";
 imprima "\nNumero de alunos com renda pessoal maior que a familiar :",
 cont2;
 imprima "\n";
fimprog
```

algoritmo 302

Uma agência de turismo quer fazer um levantamento das praias da cidade para uma programação turística de verão, sabendo-se que cada praia tem um nome e uma distancia (em km) do hotel. Criar um algoritmo que forneça os seguintes dados:

- *porcentagem de turistas nas praias próprias do hotel;*
- *a praia mais distante;*
- *nome e distância das praias não-próprias com distância do hotel maior que 10 km;*

O algoritmo acaba quando se digita @ para nome da praia.

```
prog enq41
 int a, cont;
 real dist, mdist;
 string nome, cond, mnome;
 a <-0;
 cont <- 0;
 mdist <-0.;
 imprima "\nDigite o nome da praia ou @ para terminar:";
 leia nome;
 enquanto(nome < > "@")
 { a++;
   imprima "\nQual  a distancia do hotel ?";
   leia dist;
   imprima "\nPraia P(ropria) ou N(nao propria)?  ";
```

```
    leia cond;
    enquanto(cond < > "P"&& cond < > "p"&& cond < > "N"&& cond < > "n")
    { imprima "\nAtencao! Digite somente P(ropria) ou N(ao propria)?  ";
      leia cond;
    }
    se(cond=="P"|| cond =="p")
    { cont++;    }
    senao
    {
     se(dist>10.)
     {
      imprima "\nNome da praia :", nome;
      imprima "\nDistancia hotel :", dist;
     }
   }
   se(dist > mdist)
   { mdist <- dist;  mnome <- nome; }
   imprima "\nDigite o nome da praia ou @ para terminar:";
   leia nome;
  }
  imprima "\nA porcentagem de turistas nas praias proprias foi de :",
            formatar((cont*100)/a, 2), "%."; # continuacao
  imprima "\nPraia mais distante :", mnome;
  imprima "\n";
fimprog
```

algoritmo 303

Criar um algoritmo que ajude o DETRAN a saber o total de recursos que foram arrecadados com a aplicação de multas de trânsito.

O algoritmo deve ler as seguintes informações para cada motorista:

- *número da carteira de motorista (de 1 a 4327),*
- *número de multas;*
- *valor de cada uma das multas.*

Deve ser impresso o valor da dívida para cada motorista e ao final da leitura o total de recursos arrecadados (somatório de todas as multas). O algoritmo deverá imprimir também o número da carteira do motorista que obteve o maior número de multas.

O algoritmo termina ao ler a carteira de motorista de valor 0.

```
prog enq42
 int c, cart, nmult, mcart, mnmult;
 real valor, total;
 mnmult <- 0;
 imprima "\ndigite carteira de motorista ou 0 para terminar:  ";
```

```
leia cart;
enquanto( cart < > 0)
{  total <- 0.;
  imprima "\ndigite numero de multas:  ";
  leia nmult;
  se(nmult > mnmult)
  {  mnmult <- nmult; mcart <- cart; }
  para(c <-1; c <= nmult; c++)
  { imprima "\ndigite valor da multa:  ";
    leia valor;
    total <- total + valor;
   }
    imprima "\ncarteira: ", cart;
    imprima "\nvalor a pagar: ", total;
    imprima "\ndigite carteira de motorista ou 0 para terminar:  ";
    leia cart;
}
imprima "\nnumero da carteira com maior numero de multas: ", mcart;
imprima "\n";
fimprog
```

algoritmo 304

Criar um algoritmo que leia um conjunto de valores inteiros positivos e cujo último valor é "-1". Dentre os valores lidos, o algoritmo deve imprimir:

- *O menor valor dentre os maiores que 100 e menores que 1000;*
- *A média desses valores dentre os maiores que 100 e menores que 1000;*
- *A soma desses valores dentre os maiores que 100 e menores que 1000;*
- *A soma de todos os valores lidos.*

O valor "-1" não deve ser considerado;

Se nenhum valor estiver dentro do intervalo, o algoritmo deve imprimir uma mensagem para o usuário explicando o ocorrido.

```
prog enq43
 int num, cont, tot1, tot2, menor;
 cont  <-  0;
 tot1  <-  0;
 tot2  <-  0;
 imprima "\nDigite numero ou -1 para acabar: ";
 leia num;
 menor  <-  1001;
 enquanto(num > -1)
 {
  tot2  <-  tot2 + num;
  se(num>100 && num<1000)
  {  cont ++;   tot1  <-  tot1 + num;
     se(num<menor)
```

```
      { menor  <-   num; }
   }
   imprima "\nDigite numero ou -1 para acabar: ";
   leia num;
  }
  se (cont <> 0)
  { imprima "\nMenor entre 100 e 1000: ", menor;
    imprima "\nMedia entre 100 e 1000: ", tot1/cont;}
    imprima "\nSoma entre 100 e 1000: ", tot1;
  }
  senao
  { imprima "\nNao foi digitado numero entre 100 e 1000";}
  imprima "\nSoma de todos: ", tot2;
  imprima "\n";
 fimprog
```

algoritmo 305

Criar um algoritmo que gerencie as contas correntes dos clientes do Banco Oir Cup, um banco ítalo-anglo-franco-luso-nipo-brasileiro. O algoritmo deverá ler, para cada cliente, o código do cliente (tipo inteiro), saldo anterior (tipo real) e as movimentações da conta.

Cada movimentação é composta por um código (tipo caractere, C, D ou F, indicando Crédito, Débito ou fim das movimentações deste cliente) e um valor (tipo real). Deverá ser impresso, para cada cliente, o seu código e o saldo atual (após o processamento das movimentações).

Ao final, deverá ser impresso o total de dinheiro em caixa no banco (soma dos saldos de todos os clientes) e o código do cliente que possui o maior saldo. O algoritmo se encerra quando se digita um código menor ou igual a zero .

```
prog enq44
 int n1, maiorcod;
 real maior,n2,n4,tot;
 string n3;
 tot <-  0.;
 imprima "\nEntre com o codigo do cliente ou codigo menor ou igual a 0
          para terminar: "; # continuacao
 leia n1;
 maior <- -999999999.00;
 enquanto( n1 > 0 )
 {  imprima "\nEntre com o saldo anterior: ";
    leia n2;
    imprima "\nEntre com o tipo de movimentacao C(Credito) D(Debito)
             F(fim das movimentacoes deste cliente): "; # continuacao
    leia n3;
    enquanto( n3 <> "f" && n3 <>"F" )
    {
      imprima "\nEntre com o valor da movimentacao: ";
```

```
      leia n4;
      se(n3=="c" || n3=="C")
      { n2 <- n2 + n4;  }
      senao
      {  se(n3=="d" || n3=="D")
         {  n2 <- n2 - n4;  }
      }
     imprima "\nEntre com o tipo de movimentacao C(Credito) D(Debito)
               F(fim das movimentacoes deste cliente): "; # continuacao
     leia n3;
   }
   imprima "\nseu  codigo e ", n1, "\no saldo atual  " , n2;
   tot <- tot  +  n2;
   se(n2 > maior)
   {  maiorcod <- n1;   maior <- n2;  }
   imprima "\nEntre com o codigo do cliente ou codigo menor ou igual a 0
             para terminar : "; # continuacao
   leia n1;
  }
  imprima "\no total de dinheiro em banco : " , tot;
  imprima "\n",maiorcod, " E o codigo do cliente com maior saldo: ",  maior;
  imprima "\n";
fimprog
```

algoritmo 306

A TELEMAR deseja calcular as contas telefônicas de seus assinantes através do computador.

A cobrança de seus serviços é feita da seguinte maneira:

1. *Tarifa básica*
   - *telefone residencial (código 1): R$ 23,00*
   - *telefone comercial (código 2): R$ 30,00*
2. *Serviço local:*
   - *R$ 0,10 por pulso excedente (acima de 90 pulsos)*
3. *Serviço despertador:*
   - *R$ 0,47 por vez.*

Na entrada de dados teremos:

- *código do assinante;*
- *tipo do telefone (comercial ou residencial);*
- *número de pulsos registrados para chamadas locais e número de serviços de despertador prestados.*

Criar um algoritmo que leia os dados de um conjunto de assinantes (o código do assinante igual a zero encerra a entrada de dados), calcule e imprima:

- *para cada assinante, o total de sua conta;*
- *valor da maior conta e o código do assinante que a pagou;*
- *valor médio arrecadado por assinante no mês.*

```
prog enq45
 int num,ca,tt,pl,sd,mca;
 real mconta,soma,conta;
 mconta  <-  0.;
 soma <- 0.;
 num <- 0;
 imprima "\ndigite o codigo do assinante ou 0 para terminar: ";
 leia ca;
 enquanto(ca < > 0)
 { imprima "\ndigite o tipo do telefone( 1 residencial 2 comercial): ";
   leia tt;
   imprima "\ndigite o numero de pulsos locais: ";
   leia pl;
   imprima "\ndigite o numero de servicos despertador: ";
   leia sd;
   se(tt == 1)
   { se(pl > 90)
     {conta <- 23.00 + ( (pl - 90) * 0.1) + (0.47 * sd);}
     senao
     {conta <- 23.00 + (0.47 * sd);}
   }
   senao
   {  se(pl > 90)
      {conta <- 30.00 + ( (pl - 90) * 0.1) + (0.47 * sd);}
      senao
      {conta <- 30.00 + (0.47 * sd);}
  }
  num <- num +1;
  soma <- soma + conta;
  se(conta > mconta)
   {mconta <- conta;
    mca <- ca;
   }
  imprima "\no total da conta e R$ ",conta;
  imprima "\ndigite o codigo do assinante ou 0 para terminar: ";
  leia ca;
 }
 imprima "\no valor da maior conta e R$ ",mconta, "e o codigo e: ", mca;
 imprima "\no valor medio arrecadado e R$ ",soma/num;
 imprima "\n";
fimprog
```

algoritmo 307

Criar um algoritmo que leia um conjunto de informações (nome, sexo, idade, peso e altura) dos atletas que participaram de uma olimpíada, e informar:

a) o atleta do sexo masculino mais alto;
b) a atleta do sexo feminino mais pesada;
c) a média de idade dos atletas.

Deverão ser lidos dados dos atletas até que seja digitado o nome @ para um atleta.

```
prog enq46
 int idade, cont;
 real peso, pesom, alt, altm, soma;
 string nome, sexo, nomem, nomep;
 altm <- 0.;
 pesom <- 0.;
 soma <- 0.;
 cont <- 0;
 nomem <-"";
 nomep <-"";
 imprima "\ndigite o nome do atleta ou @ para terminar: ";
 leia nome;
 enquanto(nome < > "@")
 {  cont++;
   imprima "\ndigite sexo (M / F): ";
   leia sexo;
   imprima "\ndigite idade: ";
   leia idade;
   imprima "\ndigite peso: ";
   leia peso;
   imprima "\ndigite altura: ";
   leia alt;
   soma <- soma + idade;
   se(sexo == "M" || sexo == "m" )
    {  se(alt > altm)
      { altm <-  alt; nomem <- nome;}
    }
    senao
    {   se(peso> pesom)
        { pesom <-   peso; nomep <- nome;}
    }
   imprima "\ndigite o nome do atleta ou @ para terminar: ";
   leia nome;
 }
 imprima "\nnome do atleta do sexo masculino mais alto: ", nomem;
```

```
 imprima "\nnome da atleta do sexo feminino mais pesada: ", nomep;
 imprima "\nmedia das idades dos atletas: ", soma / cont;
 imprima "\n";
fimprog
```

algoritmo 308

Criar um algoritmo que calcule quantos litros de gasolina são usados em uma viagem, sabendo que um carro faz 10 km/litro. O usuário fornecerá a velocidade do carro e o período de tempo que viaja nesta velocidade para cada trecho do percurso. Então, usando as fórmulas distância = tempo × velocidade e litros consumidos = distância/10, o algoritmo computará, para todos os valores não-negativos de velocidade, os litros de combustível consumidos. O algoritmo deverá imprimir a distância e o número de litros de combustível gastos naquele trecho. Deverá imprimir, também, o total de litros gastos na viagem. O algoritmo pára quando for digitado um valor negativo de velocidade.

```
prog enq47
 real vel, litros, soma, tempo;
 int tempohora, tempomin;
 soma <- 0.;
 imprima "Digite a velocidade(em Km/H) ou valor negativo para acabar: ";
 leia vel;
 enquanto(vel >= 0.)
 {
  imprima "\nEntre com o total de horas da viagem: ";
  leia tempohora;
  imprima "\nEntre com o total de minutos da viagem: ";
  leia tempomin;
  tempo <-tempohora + (tempomin/60);
  litros <- (vel * tempohora)/10;
  soma <- soma + litros;
  imprima "\nDistancia: ", vel * tempo, " Km";
  imprima "\nLitros gastos no percurso: ", litros;
  imprima "\nDigite a velocidade ou valor negativo para acabar: ";
  leia vel;
 }
 imprima "\nO total de litros gastos na viagem foi: ", soma, " litros";
 imprima "\n";
fimprog
```

algoritmo 309

Criar um algoritmo que calcule o imposto de renda de um grupo de contribuintes, considerando que:

a) os dados de cada contribuinte (CIC, número de dependentes e renda bruta anual) serão fornecidos pelo usuário via teclado;

b) para cada contribuinte será feito um abatimento de R$ 600 por dependente;

c) a renda líquida é obtida diminuindo-se o abatimento com os dependentes da renda bruta anual;

d) para saber quanto o contribuinte deve pagar de imposto, utiliza-se a tabela a seguir:

RENDA LÍQUIDA	*IMPOSTO*
até R$ 1000	*0% (isento)*
de R$ 1001 a R$ 5000	*15%*
acima de R$ 5000	*25%*

e) o valor de CIC igual a zero indica final de dados,

f) o algoritmo deverá imprimir, para cada contribuinte, o número do CIC e o imposto a ser pago;

g) ao final o algoritmo deverá imprimir o total do imposto arrecadado pela Receita Federal e o número de contribuintes isentos;

h) leve em consideração o fato de o primeiro CIC informado poder ser zero.

```
prog enq48
 int cic, ndep, isento;
 real rb, total, ir, totir;
 imprima "\ndigite  o seu CIC ou 0 para acabar: ";
 leia cic;
 totir  <- 0.;
 isento <-0;
 enquanto(cic > 0)
 {
    imprima "\nDigite o numero de dependentes: ";
    leia ndep;
    imprima "\nDigite sua renda bruta: ";
    leia rb;
    total <- rb - 600.00 * ndep;
    imprima "\n\nCIC: ",cic;
    se(total <= 1000.00)
    {
        imprima "\nIsento ";
        isento <- isento + 1;
     }
     senao
     {  se( total <=  5000.00 )
        {   ir <- total * 0.15;
            totir <- totir + ir;
            imprima "\nO imposto a ser pago: ", ir ;
         }
         senao
         {   ir <- total * 0.20;
```

```
            totir <- totir + ir;
            imprima "\nO imposto a ser pago  de: ", ir;
        }
    }
    imprima "\nDigite o seu CIC ou 0 para acabar: ";
    leia cic;
}
imprima "\nO total de imposto arrecadado pela receita  de: ", totir;
imprima "\nO numero de contribuintes isentos: ", isento;
imprima "\n";
fimprog
```

algoritmo 310

Foi feita uma pesquisa de audiência de canal de TV em várias casas de uma certa cidade, em um determinado dia.

Para cada casa visitada foram fornecidos o número do canal (4, 5, 7, 12) e o número de pessoas que estavam assistindo a ele naquela casa. Se a televisão estivesse desligada, nada seria anotado, ou seja, esta casa não entraria na pesquisa. Criar um algoritmo que:

- *Leia um número indeterminado de dados, isto é, o número do canal e o número de pessoas que estavam assistindo;*
- *Calcule e imprima a porcentagem de audiência em cada canal.*

Para encerrar a entrada de dados, digite o número do canal zero.

```
prog enq49
 int op,num,can4,can5,can7,can12, totent;
 can4 <- 0;
 can5 <- 0;
 can7 <- 0;
 can12 <- 0;
 totent <- 0;
 imprima "\nCanal(4/5/7/12), para terminar 0: ";
 leia op;
 enquanto(op>0)
 {
  se(op< >4 && op< >5 && op< >7 && op< >12)
  {imprima "\nopcao invalida";}
  senao
  {
   imprima "\nnumero de pessoas que estavam assistindo: ";
   leia num;
   totent <- totent + num;
   se(op == 4)
   {can4 <- can4 + num; }
   senao
```

```
    {
     se(op == 5)
     {can5 <- can5 + num; }
     senao
     {
      se(op == 7)
      {can7 <- can7 + num; }
       senao
       {
        se(op == 12)
        {can12 <- can12 + num; }
     }}}}

  imprima "\nCanal(4/5/7/12), para terminar 0: ";
  leia op;
 }
 imprima  "\nporcentagem de audiencia do canal 4: ", formatar(can4*100/totent,2);
 imprima  "\nporcentagem de audiencia do canal 5: ", formatar(can5*100/totent,2);
 imprima  "\nporcentagem de audiencia do canal 7: ", formatar(can7*100/totent,2);
 imprima  "\nporcentagem de audiencia do canal 12: ",
            formatar(can12*100/totent,2); # continuacao
 imprima "\n";
fimprog
```

algoritmo 311

Criar um algoritmo que calcule e imprima o CR do período para os alunos de programação I. Para cada aluno, o algoritmo deverá ler

- *número da matrícula;*
- *quantidade de disciplinas cursadas;*
- *notas em cada disciplina.*

Além do CR de cada aluno, o algoritmo deve imprimir o melhor CR dos alunos que cursaram 5 ou mais disciplinas.

- *fim da entrada de dados é marcado por uma matrícula inválida (matrículas válidas: de 1 a 5000);*
- *CR do aluno é igual à média aritmética de suas notas.*

```
prog enq50
 int mat, i, nd;
 real cr, nota, maior;
 imprima "\ndigite matricula (1 - 1500): ";
 leia mat;
 maior <- 0.;
 enquanto(mat >= 1 && mat <= 1500)
 { imprima "\nnumero de disciplinas: ";
```

```
  leia nd;
  cr <- 0.;
  para( i <- 1 ; i <=nd ; i++)
  { imprima "\ndigite nota: ";
    leia nota;
    cr <- cr + nota;
  }
  cr <- cr / nd;
  se( nd > 5)
  {
    se(cr > maior)
    { maior <- cr; }
  }
  imprima "\nMatricula: ", mat;
  imprima "\nCR: ", cr;
  imprima "\ndigite matricula (1 - 1500): ";
  leia mat;
 }
 imprima "\nMaior CR: ", maior;
 imprima "\n";
fimprog
```

algoritmo 312

Criar um algoritmo que possa entrar com vários números inteiros positivos até entrar -1. Imprimir todos os números e, ao final, o total de números múltiplos de 8 digitados e a média de todos os números lidos.

```
prog enq51
 int num, cont, tot1, tot2, menor;
 cont  <-  0;
 tot1  <-  0;
 tot2  <-  0;
 imprima "\nDigite numero ou -1 para acabar: ";
 leia num;
 enquanto(num < > -1)
 {
   tot1  <-  tot1 + num;
   cont ++;
   se(num % 8 == 0)
   {  tot2++; }
   imprima "\n", num;
   imprima "\nDigite numero ou -1 para acabar: ";
    leia num;
 }
 imprima "\nTotal de multiplos de 8: ", tot2;
```

```
 imprima "\nMedia dos numeros: ", tot1/cont;
 imprima "\n";
fimprog
```

algoritmo 313

Criar um algoritmo que receba a idade, a altura e o peso de várias pessoas. Calcule e imprima:

- *a quantidade de pessoas com idade superior a 50 anos;*
- *a média das alturas das pessoas com idade entre 10 e 20 anos;*
- *a porcentagem de pessoas com peso inferior a 40 quilos entre todas as pessoas analisadas.*

```
prog enq52
 real peso, altura, soma1020;
 int idade,c1020,c50,ct,c40;
 soma1020  <- 0.;
 ct  <-  0;
 c1020  <-  0;
 c50   <-  0;
 c40 <- 0;
 imprima "\nEntre a idade ou 0 para terminar: ";
 leia idade;
 enquanto(idade > 0)
 {
  ct ++;
  imprima "\nEntre altura: ";
  leia altura;
  imprima "\nEntre peso: ";
  leia peso;
  se(peso < 40.)
  { c40 ++; }
  se(idade > 50)
  { c50 ++; }
  senao
  {
    se(idade > 10 && idade < 20)
    { c1020++;
      soma1020 <- soma1020 +  idade;
    }
  }
  imprima "\nEntre a idade ou 0 para terminar: ";
  leia idade;
 }
 se(c40 > 0)
 { imprima "\nTotal com mais de 50 anos: ", c50;}
 senao
```

```
  {imprima "\nNinguem com mais de 50 anos"; }
  se(c1020 > 0)
   { imprima "\nMedia entre de 10 ate 20: ", soma1020/c1020;}
  senao
  {imprima "\nNinguem dentro da faixa 10 ate 20"; }
  se(c40 > 0)
  { imprima "\nPercentual com menos de 40 kg: ", formatar(c40 *100/ct,2);}
  senao
  {imprima "\nNinguem com menos de 40 kg"; }
  imprima "\n";
fimprog
```

algoritmo 314

Criar um algoritmo que receba o valor e o código de várias mercadorias vendidas em um determinado dia. Os códigos obedecem à lista a seguir:

'L' – limpeza
'A' – alimentação
'H' – higiene

Calcule e imprima:

- *o total vendido naquele dia, com todos os códigos juntos;*
- *o total vendido naquele dia em cada um dos códigos.*

Para encerrar a entrada de dados, digite o valor da mercadoria zero.

```
prog enq53
 string op;
 real valor, totlimp, totalim, tothig;
 totlimp <- 0.;
 totalim <- 0.;
 tothig <-0.;
 imprima "\nEntre com valor ou 0. para terminar: ";
 leia valor;
 enquanto (valor>0.)
 {
 imprima "\nEntre com H(higiene), A(alimentacao), L(limpeza) terminar: ";
 leia op;
 se(op == "L"|| op == "l")
 { totlimp <- totlimp + valor;}
 senao
 {
   se(op == "A"|| op == "a")
   { totalim <-totalim + valor;}
   senao
   {
   se(op == "H"|| op == "h")
```

```
    { tothig <-tothig + valor;}
    senao
    {imprima "\nopcao invalida";
  }}}
  imprima "\nEntre com valor ou 0. para terminar: ";
  leia valor;
  }
  imprima "\ntotal geral vendido: ", totlimp + totalim + tothig;
  imprima "\ntotal limpeza: ", totlimp;
  imprima "\ntotal alimentacao: ", totalim;
  imprima "\ntotal higiene: ", tothig;
  imprima "\n";
fimprog
```

algoritmo 315

Uma ONG vai distribuir presentes de Natal para crianças de uma comunidade carente. Para auxiliar na compra e controle dos brinquedos, criar um algoritmo em que todas as famílias serão cadastradas, tendo ou não crianças. Leia, para cada família, a quantidade de crianças do sexo feminino e a quantidade de crianças do sexo masculino; a leitura termina quando for digitado @ para cadastrar a família. Sabendo-se que um presente para menina custa R$12,00 e um presente para menino custa R$11,00, imprima:

- *O total gasto com presentes para meninos e o total para meninas;*
- *O percentual de famílias beneficiadas com presentes para seus filhos*

```
prog enq54
 int famt,famf,sf,sm, rs;
 string filhos, fam;
 famt <- 0;
 famf <- 0;
 sf <- 0;
 sm <- 0;
 imprima "\nCadastra familia(s ou qualquer caractere) ou @ para terminar: ";
 leia fam;
 enquanto(fam< >"@")
 { famt++;
  imprima "\nTem filhos (s/n) ? ";
  leia filhos;
  se(filhos =="s" || filhos =="S")
  { famf++;
    imprima "\nQuantas do sexo feminino? ";
    leia rs;
    sf<-sf + rs;
    imprima "\nQuantas do sexo masculino? ";
    leia rs;
```

```
    sm<-sm + rs;
   }
   imprima "\nCadastra familia(s ou qualquer caractere) ou @ para
             terminar: "; # continuacao
   leia fam;
 }
 imprima "\nTotal gasto com as  meninas: ", sf * 11.00;
 imprima "\nTotal gasto com meninos: ", sm * 12.00;
 imprima "\nPercentual de familias beneficiadas: ", famf*100/famt;
 imprima "\n";
fimprog
```

algoritmo 316

Criar um algoritmo que receba a idade e o estado civil (C– casado, S– solteiro, V– viúvo e D– desquitado ou separado) de várias pessoas. Calcule e imprima:

- *a quantidade de pessoas casadas;*
- *a quantidade de pessoas solteiras;*
- *a média das idades das pessoas viúvas;*
- *a porcentagem de pessoas desquitadas ou separadas dentre todas as pessoas analisadas.*

O algoritmo acaba quando se digita um número menor do que 0 para idade.

```
prog enq55
 int i, id, tcas, tsol, tdes, tviu, geral;
 real somaviu;
 string op;
 tcas <- 0;
 geral <- 0;
 tsol <- 0;
 tviu <- 0;
 somaviu <- 0.;
 tdes <- 0;
 imprima "\n Digite idade ou 0 para terminar: ";
 leia id;
 enquanto( id > 0)
 { geral++;
   imprima "\n estado civil C / S / V / D: ";
   leia op;
   se( op == "c" || op=="C" )
   { tcas++; }
   senao
    { se( op == "s" || op=="S" )
      { tsol++;}
      senao
        {  se( op == "d" ||op =="D" )
```

```
            { tdes++;}
           senao
           {  se( op == "v" ||op =="V" )
              { somaviu <- somaviu +id;
                 tviu++;}
           }
   }
      }
   imprima "\n Digite idade ou 0 para terminar: ";
   leia id;
  }
  se(tcas <> 0)
  {imprima "\ntotal de casados: ", tcas;}
  senao
  { imprima "\nnao existem casados";}
  se(tsol <> 0)
  { imprima "\ntotal de solteiros: ", tsol;}
  senao
  { imprima "\nnao existem solteiros";}
  se(tviu <> 0)
  {imprima "\nmdia da idade das viuvos: ", somaviu/tviu;}
  senao
  { imprima "\nnao existem viuvos";}
  se(tdes <> 0)
  { imprima "\nporcentagem de pessoas desquitados ou separados: ", tdes *
  100/geral;}
  senao
  { imprima "\nnao existem desquitados ou separados ";}
  imprima "\n";
fimprog
```

algoritmo 317

Criar um algoritmo que leia valores de venda de discos, CD's e fitas K7 de uma loja em um dia, desconte a comissão do vendedor (3%) e o repasse ao fornecedor (50%) para gerar o ganho da loja na venda de cada peça. O algoritmo deverá imprimir o valor da venda, a comissão, o repasse e o ganho da loja para cada venda, totalizando os valores ao final. Os dados da entrada terminam com o valor da venda, sendo igual ou menor do que 0 (zero).

```
prog enq56
  real venda,comissao,forn,loja,totvenda,totcomissao,totforn,totloja;
  imprima "\nDigite valor da venda ou 0 para terminar: ";
  leia venda;
  totvenda <- 0.;
  totcomissao <- 0.;
  totforn <- 0.;
```

```
 totloja <- 0.;
 enquanto( venda < > 0. )
 { comissao <- venda *  0.03;
   forn <- venda * 0.5;
   loja <- venda - comissao - forn;
   totvenda <- totvenda + venda;
   totcomissao <- totcomissao + comissao;
   totforn <- totforn + forn;
   totloja <- totloja + loja;
   imprima "\nvalor da venda: ",venda;
   imprima "\nvalor da comissao: ",comissao;
   imprima "\nvalor do repasse: ",forn;
   imprima "\nganho da loja: ",loja;
   imprima "\nDigite valor da venda ou 0 para terminar: ";
   leia venda;
 }
 imprima "\nTotal de vendas: ",totvenda;
 imprima "\nTotal de comissoes: ",totcomissao;
 imprima "\nTotal do repasse: ",totforn;
 imprima "\nGanho total da loja: ",totloja;
 imprima "\n";
fimprog
```

algoritmo 318

Criar um algoritmo que leia vários nomes completos e apresente somente o sobrenome (última parte de nome). O algoritmo acaba quando se digita @ para o nome.

Exemplo: *Entrada: Anita Luiza Maciel Lopes* *Saída: Lopes*
Entrada: Carlos Augusto Garcia de Assis *Saída: Assis*

```
prog enq57
 string nome;
 int c;
 imprima "\nDigite nome completo ou @ para terminar: ";
 leia nome;
 enquanto(nome < >"@")
 {
  c <- strtam(nome)-1;
  enquanto(c>=0 && strelem(nome,c)<>"ƀ")
  {c--;}
  c++;
  imprima "\n",strnresto(nome,c);
  imprima "\nDigite nome completo ou @ para terminar: ";
  leia nome;
 }
 imprima "\n";
fimprog
```

algoritmo 319

Criar um algoritmo que leia vários nomes completos e apresente quantos subnomes possui cada nome lido.

Exemplo: *Entrada: Anita Luiza Maciel Lopes* *Saída: 4*
Entrada: Carlos Augusto Garcia de Assis *Saída: 5*

```
prog enq58
 string nome;
 int c, cont;
 imprima "\nDigite nome completo ou @ para terminar: ";
 leia nome;
 enquanto(nome <>"@")
 { cont <-0;
  c <- strtam(nome)-1;
  enquanto(c >= 0)
  { se(strelem(nome,c)== "ƀ")
    {cont++;}
    c--;
  }
  imprima "\ntotal de sub-nomes: ",cont+1;
  imprima "\n\nDigite nome completo ou @ para terminar: ";
  leia nome;
 }
 imprima "\n";
fimprog
```

algoritmo 320

Criar um algoritmo que calcule e imprima a média e a variância da notas de todos os alunos inscritos em Programação I. As notas devem ser lidas via teclado. O fim da leitura é marcado por uma nota negativa. A média e a variância são calculadas da seguinte forma:

- *média = Σ notas/número de alunos*
- *variância = (Σ (notas ** 2)/número de alunos) – media **2*

```
prog enq59
 int cont;
 real nota, snota, qsnota, media, qmedia, variancia;
 cont <- 0;
 qsnota <- 0.;
 snota <- 0.;
 imprima "\ndigite nota ( 0 - 10): ";
 leia nota;
 enquanto(nota >= 0.)
 {
```

```
  cont++;
  snota <- snota + nota;
  qsnota <- qsnota + nota ** 2;
  imprima "\ndigite nota ( 0 - 10): ";
  leia nota;
 }
 media <- snota / cont;
 variancia <- qsnota / cont - media ** 2;
 imprima "\nMedia: ", media;
 imprima "\nVariancia: ", variancia;
 imprima "\n";
fimprog
```

algoritmo 321

Criar um algoritmo que leia as seguintes informações sobre a turma:

nº da turma
nº de aulas dadas
nº de alunos inscritos

Deve ser lido também para cada aluno inscrito o seu nº de faltas.

O algoritmo deve imprimir o percentual de faltas para cada aluno e ao final do processamento o total de alunos reprovados por falta (mais de 25% de falta).

↳ *O algoritmo acaba quando se digita * para a turma.*

```
prog enq60
 real  percfalta;
 string turma;
 int soma, nalunos, naula, nfaltas, aluno, i;
 soma <- 0;
 imprima "\ndigite turma ou * para acabar: ";
 leia  turma;
 enquanto(turma < > "*")
 { imprima "\ndigite numero de alunos: ";
  leia nalunos ;
  imprima "\ndigite numero de aulas dadas: ";
  leia naula ;
  para( i <- 1; i <= nalunos; i++)
  {  imprima "\n\ndigite numero de faltas do ", i," aluno: ";
     leia nfaltas;
     percfalta <- 100 * nfaltas / naula;
     se(percfalta >= 25.)
     { soma ++; }
     imprima "\nPercentual de faltas: ", formatar(percfalta,2), "%";
  }
```

```
  imprima "\ndigite turma ou * para acabar: ";
  leia  turma;
 }
 imprima "\ntotal de alunos reprovados por falta: ",soma;
 imprima "\n";
fimprog
```

algoritmo 322

Uma loja utiliza os seguintes códigos para as transações de cada dia:

"v" – para compras à vista

"p" – para compras a prazo

É dada uma lista de transações contendo o valor de cada compra e o respectivo código da transação. Criar um algoritmo que calcule e imprima:

- *valor total das compras à vista;*
- *valor total das compras a prazo;*
- *valor total das compras efetuadas;*
- *valor a receber pelas compras a prazo, isto é, primeira parcela, sabendo-se que as compras serão pagas em três vezes.*

O número de transações efetuadas, por dia, deverá ser digitado.

Deverá haver trecho de proteção para entrada do código de tal maneira que só aceite as letras V e P, maiúscula ou minúscula.

```
prog enq61
 real val, valvista, valprazo, valtotprazo;
 int n, i;
 string op;
 valvista <-  0.;
 valprazo <- 0.;
 valtotprazo <- 0.;
 imprima "\nquantidade de transacoess efetuadas neste dia: ";
 leia n ;
 para( i <-  1; i <= n ; i++)
 {   imprima "\ndigite valor: ";
    leia val;
    imprima "\ndigite codigo: v (a vista)  p ( a prazo ): ";
    leia op;
    # trecho de protecao para o codigo
    enquanto(  op < > "V" &&  op < > "P" && op < > "v" &&  op < > "p")
    {  imprima "\nTIPO DE CODIGO INVALIDO. Digite codigo (v / p ): ";
       leia op;
    }
    se(op == "V" || op ==  "v" )
    {  valvista <- valvista + val;}
```

```
    senao
    {   valprazo <- valprazo + val / 3;
        valtotprazo <- valtotprazo + val ;}
 }
 imprima "\n total  vista: ", valvista;
 imprima "\n total a prazo: ", valtotprazo;
 imprima "\n valor total de compras: ", valvista + valtotprazo;
 imprima "\n valor total a receber  da 1 parcela: ", valprazo;
 imprima "\n";
fimprog
```

algoritmo 323

Um clube com capacidade máxima para 2.000 pessoas em seu salão de festas, organizou um baile em que foi permitida a entrada de sócios e não-sócios cobrando os seguintes valores por cada ingresso:

SÓCIO	*R$10,00*
NÃO-SÓCIO	*R$20,00*

Criar um algoritmo que leia as informações sobre ingressos vendidos (tipo "sócio" ou "não-sócio") até que seja digitado o valor –999 ou que todos os ingressos sejam vendidos e imprima:

- *A quantidade de ingressos vendidos para sócios e a quantidade para não-sócios;*
- *O percentual de ingressos para sócios em relação ao total geral de ingressos vendidos;*
- *O valor em Reais recebido de sócios, de não-sócios e o total arrecadado no baile.*

```
prog enq62
 string tipo;
 int qtd, qtdsocio, qtdnaosocio, qtdtotal;
 real percsoc, valtotal;
 qtdsocio <- 0;
 qtdnaosocio <- 0;
 qtdtotal <- 0;
 imprima "\nDigite a quantidade ou -999 para terminar: ";
 leia qtd;
 enquanto( qtd < > -999 && qtdtotal < 2000 )
 { imprima "\n Digite o tipo de ingresso S ou NS: ";
   leia tipo;
   se(tipo == "S" || tipo == "s")
   { qtdsocio <- qtdsocio + qtd;
     qtdtotal <- qtdtotal + qtd;
   }
   senao
```

```
  { se(tipo == "NS" || tipo == "ns")
    { qtdnaosocio <- qtdnaosocio + qtd;
      qtdtotal <- qtdtotal + qtd;
    }
    senao
    { imprima "\n Tipo Invalido!";
    }
  }
  imprima "\nDigite a quantidade ou -999 para terminar: ";
  leia qtd;
 }
 imprima "\nSocio = ", qtdsocio;
 imprima "\nNao Socio = ", qtdnaosocio;
 percsoc <- qtdsocio * 100/ qtdtotal;
 imprima "\nPercentual Socio =", percsoc;
 imprima "\nValor Socio = ",qtdsocio * 10.00;
 imprima "\nValor Nao Socio = ",qtdnaosocio * 20.00;
 valtotal <- qtdsocio * 10.00 + qtdnaosocio * 20.00;
 imprima "\nValor total = ",valtotal;
 imprima "\n";
fimprog
```

algoritmo 324

Uma empresa classifica seus funcionários em três níveis de acordo com um índice de produtividade. São eles: (1) Excelente, (2) Bom e (3) Regular. Cada nível acrescenta um abono ao salário base do funcionário, de acordo com a seguinte tabela:

Excelente	*80% do salário base*
Bom	*50% do salário base*
Regular	*30% do salário base*

O algoritmo deve ler a matrícula do funcionário, seu salário base, seu nível de abono e imprimir o salário a ser pago. O algoritmo deve fornecer também a matrícula do funcionário de maior abono e a média do abono para os funcionários classificados como "Regular".

O algoritmo termina ao ler um valor de matrícula negativo.

```
prog enq63
 int mmat,cont,mat,n;
 real sb,ab,ns,mab,reg;
 mab <- 0.;
 cont <- 0;
 reg <- 0.;
 imprima "\ndigite a matricula ou numero negativo para acabar: ";
 leia mat;
 enquanto(mat >= 0)
```

```
  { imprima "\ndigite o salario base com ponto: ";
    leia sb;
    imprima "\ndigite o nivel: 1- excelente  2- bom  3- regular: ";
    leia n;
    se(n == 1)
    { ab <- sb * 0.8; ns <- sb + ab;}
    senao
    {se(n == 2)
     {ab <- sb * 0.5; ns <- sb + ab;}
     senao
     { ab <- sb * 0.3; ns <- sb + ab; cont ++; reg <- reg + ab; }
     }
     se(ab > mab)
     { mab <- ab; mmat <- mat; }
     imprima "\no salario a ser pago e ",ns;
     imprima "\n\ndigite a matricula ou numero negativo para acabar: ";
     leia mat;
  }
  imprima "\na matricula de maior abono e ",mmat;
  se( cont <> 0 )
  { imprima "\na media de abono de funcionario regular e ",reg/cont;}
  senao
  { imprima :\nNenhum funcionario regular e ";}
  imprima "\n";
fimprog
```

algoritmo 325

Criar um algoritmo que receba:

- *O valor do salário mínimo;*
- *O número de horas trabalhadas de vários funcionários (até digitar -1 para horas trabalhadas);*
- *O número de dependentes de cada funcionário;*
- *Quantidade de horas extras trabalhadas.*

Calcular e imprimir o salário a receber dos funcionários, tendo em vista as regras a seguir:

- *O valor da hora trabalhada é igual a 1/10 do salário mínimo;*
- *O salário do mês é igual ao número de horas trabalhadas vezes o valor da hora trabalhada;*
- *Para cada dependente acréscimo de 32 reais;*
- *Para cada hora extra trabalhada acréscimo de 50% ao valor da hora trabalhada;*

- *O salário bruto é igual ao salário do mês mais os valores dos dependentes e valores das horas extras;*
- *O desconto do imposto de renda retido na fonte segue a tabela a seguir:*
- *O salário líquido é igual ao salário bruto menos o IRRF;*

IRRF	*Salário Bruto*
Isento	*Até 900*
10%	*Acima de 900 até 1500*
20%	*Superior a 1500*

- *A gratificação segue a tabela a seguir:*

Salário Líquido	*Bonificação*
Até 900	*100 reais*
Superiores a 900	*50 reais*

- *O salário a receber do funcionário é igual ao salário líquido mais a gratificação.*

```
prog enq64
 real salm, salb, salr, irrf, salq, salmes, vh;
 int he, ht, dep;
 imprima "\nsalario minimo: ";
 leia salm;
 vh <- salm /10;
 imprima "\nhoras trabalhadas ou -1 para terminar: " ;
 leia ht;
 enquanto(  ht < > -1)
 { imprima "\nnumero de dependentes: ";
   leia dep;
   imprima "\nnumero de horas extras: ";
   leia he;
   salmes <- vh * ht;
   imprima "\nsalario mes: ",salmes;
   salb <- salmes + 32 * dep + vh * he * 1.5;
   se( salb > 1500. )
   { irrf <- salb * 0.20;}
   senao
   { se(salb > 900.)
     { irrf <- salb * 0.10;}
     senao
     { irrf <- salb * 0.;}
   }
   salq <- salb - irrf;
   se( salq >= 900.)
   { salr <- salq + 50;}
   senao
   { salr <- salq + 100;}
```

```
  imprima "\nsalario a receber: ",salr;
  imprima "\nhoras trabalhadas ou -1 para terminar: " ;
  leia ht;
 }
 imprima "\n";
fimprog
```

algoritmo 326

Uma empresa decidiu fazer um levantamento em relação aos candidatos que se apresentarem para preenchimento de vagas no seu quadro de funcionários. Supondo que você seja o programador dessa empresa, criar um algoritmo que leia para cada candidato a idade, o sexo (M ou F) e a experiência no serviço (S ou N). Para encerrar a entrada de dados, digite zero para a idade. Calcule e escreva:

- *o número de candidatos do sexo feminino;*
- *o número de candidatos do sexo masculino;*
- *a idade média dos homens que já têm experiência no serviço;*
- *a porcentagem dos homens com mais de 45 anos entre o total dos homens;*
- *o número de mulheres com idade inferior a 35 anos e com experiência no serviço;*
- *a menor idade entre as mulheres que já têm experiência no serviço.*

```
prog enq65
 int idade,contpessoas, contfem, contmasc,soma, contexph, conthom;
 int contexpm, contexpmt, menor; string sexo, expe;
 contpessoas <- 0; contfem <- 0; contmasc <- 0; soma <- 0;
 contexph <- 0; conthom <- 0; contexpm <- 0; contexpmt <- 0;
 imprima "\ndigite idade ou negativo para acabar: "; leia idade;
 menor <- 200;
 enquanto(idade > 0)
 { imprima "\ndigite sexo( M / F): "; leia sexo;
   imprima "\ndigite experiencia(S / N): "; leia expe;
   contpessoas++;
   se(sexo == "f" || sexo == "F")
   { contfem++;
     se(expe == "s" || expe == "S")
     { contexpmt++;
      se(idade < menor)
      { menor <- idade;}
      se(idade < 35)
      {contexpm++;}
     }
   }
   senao
   { contmasc++;
```

```
    se(expe == "s" || expe == "S")
    { contexph++; soma <- soma + idade; }
      se(idade > 45)
      { conthom++; }
   }
   imprima "\ndigite idade ou negativo para acabar: "; leia idade;
}
se( contmasc <> 0 )
{ imprima "\ncandidatos do sexo masculino: ", contmasc;
  imprima "\n% dos homens com mais de 45 anos entre o total
            dos homens: ",conthom*100/contmasc; # continuacao
  se( contexph <> 0 )
  { imprima "\nidade media dos homens que ja tem experiencia no
              servico: ", soma/contexph; } # continuacao
  senao
  { imprima "\nNenhum homem com experiencia no servico";}
}
senao
{ imprima "\nNenhum homem";}
se( contfem <> 0 )
{ imprima "\ncandidatas do sexo feminino: ", contfem;
  se( contexpmt <> 0 )
  { imprima "\nmulheres experientes com idade < 35: ", contexpm;
    imprima "\nmenor idade das mulheres experientes: ", menor;
  senao
  { imprima "\nNenhuma mulher com experiencia no servico";}
 }
 senao
 { imprima "\nNenhuma mulher";}
 imprima "\n";
fimprog
```

algoritmo 327

Criar um algoritmo que controle o saldo bancário de um cliente. O algoritmo lê o valor do saldo anterior e em seguida lê as operações realizadas na conta. As operações podem ser as seguintes:

Saque em dinheiro (código = 10)
Depósito (código = 33)
Pagamento de cheque (código = 4)

O algoritmo lê o código das operações e realiza as atualizações na conta, imprimindo uma mensagem ao usuário caso seu saldo se torne negativo.

O algoritmo deverá continuar a leitura ate que o código de operação seja zero. Códigos diferentes dos definidos devem ser ignorados.

Ao final do processamento o algoritmo deverá imprimir o saldo atual do cliente.

```
prog enq66
 real saldo, retira, val, deposito;
 int cod;
 imprima "\ndigite saldo: ";
 leia saldo;
 imprima "\ndigite codigo: Saque em dinheiro(10) Deposito(33) Pagamento
          de cheque(4) Finalizar (0): "; # continuacao
 leia cod;
 enquanto(cod < > 0)
 {  se(cod == 10)
   {  imprima "\ndigite valor da retirada: ";
      leia retira;
      saldo <- saldo - retira;
   }
   senao
   { se(cod == 33)
     { imprima "\ndigite valor do deposito: ";
       leia deposito;
       saldo <- saldo + deposito;
     }
     senao
     { se(cod == 4)
       { imprima "Entre com o valor do cheque: ";
         leia val;
          saldo <- saldo - val;
       }
     }
  }
  se(saldo < 0. )
  { imprima "\nSaldo negativo de: ", saldo; }
  imprima "\ndigite codigo: Saque em dinheiro(10) Deposito(33) Pagamento
           de cheque(4) Finalizar (0): "; # continuacao
  leia cod;
 }
 imprima "\nSaldo atual = ", saldo;
 imprima "\n";
fimprog
```

algoritmo 328

Criar um algoritmo que permita a uma empresa atacadista de cimento controlar os pedidos de compra e o estoque do produto. O algoritmo inicialmente lerá do teclado a quantidade de sacos de cimento disponíveis no estoque da compra (tipo int) e o preço de cada saco (tipo real). Em seguida, o algoritmo processará os pedidos de compra dos clientes da empresa.

As informações a serem lidas do teclado, para cada pedido, são as seguintes:

- *Código do cliente (tipo int)*
- *Quantidade de sacos de cimento a serem comprados (tipo int)*

Um pedido só poderá ser aceito se a quantidade de sacos disponíveis no estoque for maior ou igual à quantidade de sacos do pedido e se a quantidade de sacos pedidos não ultrapassar 10% do total de sacos disponíveis no estoque.

Se o pedido for aceito, o algoritmo deverá imprimir o código do cliente, a quantidade de sacos pedidos e o valor do pedido, além de subtrair do estoque a quantidade de sacos vendidos.

Se o pedido for rejeitado, as mensagens "Estoque Insuficiente" ou "Ultrapassado o máximo permitido" deverão ser exibidas no vídeo, conforme o motivo da rejeição do pedido.

Ao final dos pedidos, que será detectado quando o código do cliente for zero ou o estoque mínimo de cem sacos for atingido, deverá ser impressa a quantidade de sacos em estoque.

```
prog enq67
 int qsde, codcli,p, qs;
 real ps, vp;
 imprima "\nentre com a quantidade de sacos disponiveis em estoque: ";
 leia qsde;
 imprima "\nentre com o preco de cada saco: ";
 leia ps;
 p <- realint(qsde * 0.1);
 imprima "\nentre com o codigo do cliente ou 0 para terminar: ";
 leia codcli;
 enquanto( codcli < > 0 &&  qsde > 100)
 {  imprima "\nentre com a quantidade de sacos a serem comprados: ";
   leia qs;
   vp <- qs * ps;
   se(qsde >= qs && qs <= p)
   { imprima "\ncodigo do cliente = ", codcli;
     imprima "\nquantidade de sacos pedidos = ", qs;
     imprima "\nvalor do pedido = ", vp;
      qsde <- qsde - qs;
   }
  senao
  {  se(qsde < qs)
     { imprima "\nestoque insuficiente"; }
     senao
     {  se( qs > p)
        { imprima "\nultrapassado o maximo permitido"; }
     }
  }
  imprima "\n\nentre com o codigo do cliente ou 0 para terminar: ";
  leia codcli;
 }
 imprima "\nquantidade de sacos em estoque = ", qsde;
 imprima "\n";
fimprog
```

algoritmo 329

Em uma eleição presidencial, existem quatro candidatos. Os votos são informados através de código. Os dados utilizados para a escrutinagem obedecem à seguinte codificação:

- *1, 2, 3, 4 = voto para os respectivos candidatos;*
- *5 = voto nulo;*
- *6 = voto em branco;*

Elaborar um algoritmo que calcule e imprima:

- *total de votos para cada candidato:*
- *total de votos nulos;*
- *total de votos em branco,*
- *percentual dos votos em branco e nulos sobre o total*

```
prog enq68
 int voto, cand1, cand2, cand3, cand4, vn, vb, total;
 cand1 <- 0;
 cand2 <- 0;
 cand3 <- 0;
 cand4 <- 0;
 vn <- 0;
 vb <- 0;
 total <- 0;
 imprima "\ndigite candidadtos (1 / 2 / 3 /4) 5(voto nulo ) 6(voto em
           branco) 7(finalizar): "; # continuacao
 leia voto;
 enquanto( voto < > 7)
 {
 enquanto( voto < 1 || voto >7 )
 { imprima "\nVOTO INVALIDO.";
   imprima "\ndigite candidato (1 / 2 / 3 /4) 5(voto nulo ) 6(voto em
             branco) 7(finalizar): "; # continuacao
   leia voto;
 }
  total++;
  se(voto == 1)
  { cand1++; }
  senao
  { se(voto == 2)
    { cand2++; }
    senao
    { se(voto == 3)
      { cand3++; }
      senao
      { se(voto == 4)
```

```
      { cand4++; }
      senao
    { se( voto == 5 )
      { vn++; }
      senao
      { se( voto == 6 )
       { vb++; }
     }
    }
   }
  }
 }
imprima "\ndigite candidadto (1 / 2 / 3 /4) 5(voto nulo ) 6(voto em
          branco) 7(finalizar): "; # continuacao
leia voto;
}
imprima "\ntotal de votos do candidato 1: ", cand1;
imprima "\ntotal de votos do candidato 2: ", cand2;
imprima "\ntotal de votos do candidato 3: ", cand3;
imprima "\ntotal de votos do candidato 4: ", cand4;
imprima "\ntotal de votos nulos: ", vn;
imprima "\ntotal de votos em branco: ", vb;
imprima "\npercentual dos votos em branco e nulos sobre o total: ",
          formatar((vn +vb)*100 / total, 2), "%"; # continuacao
imprima "\n";
fimprog
```

algoritmo 330

A prefeitura de Arraial do Cabo realizou uma pesquisa entre vários habitantes da cidade, coletando dados sobre o rendimento familiar e o número de filhos menores de cada família. A prefeitura deseja saber:

- *média dos rendimentos da população;*
- *média do número de filhos;*
- *percentual de famílias com rendimento igual ou superior a $100 dólares.*

↳ *O fim da leitura se dará com a entrada de um rendimento negativo ou 0.*

```
prog  enq69
 int contt,contf, contd, fil;
 real rend, soma, dolar;
 soma <- 0.;
 contt <- 0;
 contf <- 0;
 contd <- 0;
 imprima "\ndigite valor do dolar em reais: ";
 leia dolar;
```

```
imprima "\ndigite rendimento familiar. Para terminar: 0 ou numero
          negativo: "; # continuacao
leia rend;
enquanto(rend > 0.)
{ contt++;
  soma <- soma + rend;
  se(rend >= 100*dolar)
  {contd++;}
  imprima "\ndigite numero de filhos: ";
  leia fil;
  contf <- contf + fil;
  imprima "\ndigite rendimento familiar. Para terminar, 0 ou numero
            negativo: "; # continuacao
  leia rend;
}
imprima  "\nmedia dos rendimentos da populacao: ", soma / contt;
imprima "\nmedia do numero de filhos: ", contf / contt;
imprima "\npercentual de familias com rendimento igual ou superior a
          $100 dolares: ",formatar( contd *100/contt,2),"%"; # continuacao
imprima "\n";
fimprog
```

algoritmo 331

Um marciano chegou a uma floresta e se escondeu atrás de uma das 100 árvores quando viu um caçador. O caçador só tinha cinco balas em sua espingarda. Cada vez que ele atirava, e não acertava, é claro, o marciano dizia: estou mais à direita ou mais à esquerda. Se o caçador não conseguir acertar o marciano, ele será levado para Marte. Implementar este jogo para dois jogadores, onde um escolhe a árvore em que o marciano irá se esconder, e o outro tenta acertar.

```
prog  enq70
 int arv, cont, resp;
 cont <- 0;
 imprima "\nescolha uma arvore de 1 - 100:  ";
 leia arv;
 faca
 {  imprima "\nqual o numero da arvore que o marciano esta escondido?
             (1 - 100): "; # continuacao
   leia resp;
   cont++;
   se(resp == arv)
   { imprima "\nVoce me acertou! Vou morrer. ";}
   senao
   { se(resp > arv)
     { imprima "\nHA! HA! HA! Estou mais a esquerda ";}
     senao
     { imprima "\nHA! HA! HA! Estou mais a direita ";}
   }
```

```
}
enquanto( resp < > arv && cont <5)
se(resp < > arv)
{ imprima "\nCacador, voce vai morrer! ";}
imprima "\n";
fimprog
```

algoritmo 332

O DCE de uma universidade resolveu fazer uma pesquisa com os alunos para descobrir quantos alunos tinham usado a xerox do DCE no último mês.

Criar um algoritmo que determine:

- *total de alunos que fizeram uso da xerox;*
- *percentual de alunos que utilizou menos de 5 vezes;*
- *percentual de alunos que utilizou entre 5 e 10 vezes inclusive;*
- *percentual de alunos que utilizou mais de 10 vezes.*

O fim da leitura será determinado quando se digitar F.

```
prog  enq71
 int cont5,cont510, cont10, cont, vezes;
 string resp;
 cont5 <- 0;
 cont510 <- 0;
 cont10 <- 0;
 cont <- 0;
 imprima "\nvoce fez uso da xerox?( S / N / F):  ";
 leia resp;
 enquanto( resp <> "F" && resp <> "f")
 { se(resp == "S" || resp == "s")
   { cont++;
     imprima "\nquantas vezes?  ";
     leia vezes;
     se( vezes < 5 )
     { cont5++; }
     senao
     { se( vezes < 10 )
       { cont510++; }
       senao
       { cont10++; }
     }
   }
 imprima "\nvoce fez uso da xerox?( S / N / F):  ";
 leia resp;
```

```
 }
 imprima  "\ntotal de alunos que fizeram uso da xerox: ", cont;
 imprima "\npercentual de alunos que utilizou menos de 5 vezes: ",
          formatar( cont5 *100/cont,2), "%"; # continuacao
 imprima "\npercentual de alunos que utilizou entre 5 e 10 vezes
          formatar( cont510 *100/cont,2), "%"; # continuacao
 imprima "\npercentual de alunos que  utilizou mais de 10 vezes: ",
          formatar( cont10 *100/cont,2), "%"; # continuacao
 imprima "\n";
fimprog
```

algoritmo 333

Sabendo-se que uma empresa que patrocina uma equipe de vôlei paga aos seus jogadores, além do salário, um valor adicional ao salário mensal (bicho) que é função da produtividade de cada um e que essa produtividade é paga de acordo com a tabela a seguir:

Classe	*Nível*	*Valor adicional*
1	*excelente*	*+100%*
2	*bom*	*+80%*
3	*médio*	*+50%*
4	*regular*	*+30%*
5	*precisa treinar mais*	*+10%*
6	*te cuida*	*+5%*
7	*tsktsk*	*nada*

Criar um algoritmo que entre com o salário e o código da classe de todos os jogadores, calcule e imprima o seu salário final e o nome da sua classe (nível). O algoritmo acaba quando se digita um número fora do intervalo da faixa de 1-7.

```
prog enq72
 #nao roda na versao2 do UAL
 int cod;
 real sal;
 imprima "\ndigite codigo (1 - 7): ";
 leia cod;
 enquanto( cod > 0 && cod < 8 )
 { imprima "\ndigite salario: ";
  leia sal;
  escolha( cod )
  {
    caso 1:
    imprima "\nseu nivel e excelente. Seu salario com bonus: ", sal *2;}
    pare;
```

```
   caso 2:
   imprima "\nseu nivel e bom. Seu salario com bonus: ", sal *1.8;}
   pare;
   caso 3:
   imprima "\nseu nivel e medio. Seu salario com bonus: ", sal *1.5;}
   pare;
   caso 4:
   imprima "\nseu nivel e regular. Seu salario com bonus: ", sal *1.3;}
   pare;
   caso 5:
   imprima "\nprecisa treinar mais. Seu salario com bonus: ", sal *1.10;}
   pare;
   caso 6:
   imprima "\nte cuida. Seu salario com bonus: ", sal *1.05;}
   pare;
   senao
   imprima "\ntsktsk. Seu salario nao tem bonus: ", sal;}
  }
  imprima "\ndigite codigo (1 - 7): ";
  leia cod;
 }
 imprima "\n";
fimprog
```

algoritmo 334

Num frigorífico existem vários bois. Cada boi traz preso no seu pescoço um cartão contendo um número de identificação e seu peso. Implementar um algoritmo que escreva o número e o peso do boi mais gordo e do boi mais magro. O algoritmo acaba quando se digita um número menor ou igual a zero para identificação. (não é necessário armazenar os dados de todos os bois).

```
prog  enq73
 int iden, maioriden, menoriden;
 real peso, maiorpeso, menorpeso;
 maiorpeso <- 0.;
 menorpeso <- 10000000.;
 imprima "\ndigite identificacao do boi ou, para terminar, 0 ou numero
          negativo: "; # continuacao
 leia iden;
 enquanto( iden > 0 )
 { imprima "\ndigite peso do boi: ";
   leia peso;
   se( peso > maiorpeso )
   { maiorpeso <- peso; maioriden <- iden; }
   senao
   { se( peso < menorpeso )
     { menorpeso <- peso; menoriden <- iden; }
```

```
    }
    imprima "\ndigite identificacao do boi ou, para terminar, 0 ou numero
              negativo: "; # continuacao
    leia iden;
  }
  imprima  "\nmaior peso ", maiorpeso, " e do boi cuja identificacao e: ",
            maioriden; # continuacao
  imprima  "\nmenor peso ", menorpeso, " e do boi cuja identificacao e: ",
            menoriden; # continuacao
  imprima "\n";
fimprog
```

algoritmo 335

A prefeitura de Pelotas resolveu fazer uma pesquisa sobre algumas características físicas da sua população e coletou os seguintes dados, referentes a cada habitante, para serem analisados:

- *sexo (masculino, feminino);*
- *cor dos olhos (azuis, verdes, castanhos);*
- *cor dos cabelos (louros, castanhos, pretos);*
- *idade em anos.*

Para cada habitante, foi preenchida uma ficha com os dados acima. Criar um algoritmo que determine e imprima:

a) a maior idade dos habitantes;

b) porcentagem de indivíduos do sexo feminino cuja idade esteja entre 18 e 35 anos, inclusive, e que tenha olhos verdes e cabelos louros.

↳ *O algoritmo se encerra quando se digita –1 para idade.*

```
prog  enq74
 string olhos, cabelos, sexo;
 int idade, maior, contf, cont;
 maior <- 0;
 contf <- 0;
 cont <- 0;
 imprima "\ndigite idade ou -1 para terminar: ";
 leia idade;
 enquanto( idade <> -1 )
 { cont++;
   imprima "\ndigite sexo ( M / F): ";
   leia sexo;
   imprima "\ndigite cor dos olhos(azuis, verdes, castanhos) : ";
   leia olhos;
```

```
  imprima "\ndigite cor dos cabelos (louros, castanhos, pretos): ";
  leia cabelos;
  se( idade > maior )
  { maior <- idade; }
  se( idade >= 18 && idade <=35 && ( cabelos == "louros" || cabelos ==
     "LOUROS" || cabelos == "Louros") && ( olhos == "verdes" || olhos ==
     "VERDES" || olhos == "verdes") ) # continuacao
  { contf++; }
  imprima "\ndigite idade ou -1 para terminar: ";
  leia idade;
 }
 imprima "\nmaior idade entre os habitantes: ", maior;
 imprima "\n porcentagem de individuos do sexo feminino cuja idade esteja
          entre 18 e 35 anos,inclusive, e que tenham olhos verdes e
          cabelos louros: ", contf * 100 /cont; # continuacao
 imprima "\n";
fimprog
```

algoritmo 336

Uma escola dá desconto de 10% para o segundo filho, 20% para o terceiro filho, 30% para o quarto, 40% para o quinto e assim sucessivamente. As mensalidades são diferentes conforme tabela a seguir:

Pré-escola	*R$ 300,00*
1o ciclo do Ensino Fundamental	*R$ 400,00*
2o ciclo do Ensino Fundamental	*R$ 500,00*
Ensino Médio	*R$ 600,00*

Criar um algoritmo que possa entrar com número de filhos e escolaridade de cada família e imprima o valor total a ser pago por cada família e o total arrecadado pela escola.

↳ *O algoritmo se encerra quando se digita 0 para número de filhos.*

```
prog  enq75
 int fil, esc, c;
 real prest, total, totalesc;
 totalesc <- 0.;
 imprima "\ntotal de filhos ou 0 para terminar: ";
 leia fil;
 enquanto( fil  > 0 )
 { imprima "\ndigite escolaridade do 1o filho: 1 pre-escola  2 1o ciclo
            do EF 3  2o ciclo do EF 4 EM: "; # continuacao
   leia esc;
   se(esc ==1 )
```

```
  { total <- 300.00; }
  senao
  { se(esc ==2 )
    { total<- 400.00; }
    senao
    { se(esc == 3 )
      { total <- 500.00; }
      senao
      { se(esc == 4 )
        { total <- 600.00 ; }
      }
    }
   }
   se( fil > 1 )
   { para(c <-1; c <= fil -1; c++)
     { imprima "\ndigite escolaridade do ",c+1, "o filho: 1 pre-escola  2
                 1o ciclo do EF 3  2o ciclo do EF  4 EM:  "; # continuacao
       leia esc;
       se(esc ==1 )
       { prest <- 300.00 - (300.00 * c/10); }
       senao
       { se(esc ==2 )
         { prest <- 400.00 - (400.00 * c/10); }
         senao
         { se(esc == 3 )
           { prest <- 500.00 - (500.00 * c/10); }
           senao
           { se(esc == 4 )
             { prest <- 600.00 - (600.00 * c/10); }
           }
         }
       }
       total <- total + prest;
     }
    }
    imprima "\ntotal a ser pago pela familia: ", total;
    totalesc <- totalesc + total;
    imprima "\ntotal de filhos ou 0 para terminar: ";
    leia fil;
 }
 imprima "\ntotal arrecadado pela escola: ", totalesc;
 imprima "\n";
fimprog
```

algoritmo 337

O prefeito de Arraial do Cabo está preocupado com o número de inadimplentes do imposto predial.

Criar um algoritmo que calcule e imprima o valor da multa a ser paga, considerando que devem ser fornecidos os seguintes dados de cada imóvel:

- *número do registro na prefeitura;*
- *valor do imposto;*
- *número de meses em atraso*

As multas devem ser calculadas a partir do valor do imposto e de acordo com a seguinte tabela:

Valor do Imposto	*% por mês em atraso*
até R$ 500,00	*1 %*
de R$ 500,01 a R$ 1800,00	*2 %*
de R$ 1800,01 a R$ 5000,00	*4 %*
de R$ 5000,01 a R$ 12000,00	*7 %*
acima de R$ 12.000,00	*10 %*

O algoritmo deve ler dados até que seja digitado um número de registro de imóvel igual a 0 (zero). Ao final de cada imóvel lido, o algoritmo deve apresentar o nº do registro do imóvel, valor do imposto, meses em atraso, a multa a ser paga e o valor total.

```
prog enq76
 int reg, meses;
 real imposto, multa;
 imprima "\ndigite numero do registro do imovel ou 0 para terminar: ";
 leia reg;
 enquanto(reg > 0)
 { imprima "\nvalor do imposto a ser pago: ";
   leia imposto;
   imprima "\nnumero de meses em atraso: ";
   leia meses;
   se( imposto <= 500.00)
   { multa <- imposto * 0.01 * meses;}
   senao
   { se( imposto <= 1800.00)
     { multa <- imposto * 0.02 * meses;}
     senao
     { se( imposto <= 5000.00)
       { multa <- imposto * 0.04 * meses;}
       senao
       { se( imposto <= 12000.00)
         { multa <- imposto * 0.07 * meses;}
         senao
         { multa <- imposto * 0.1 * meses;}
       }
     }
   }
```

```
    imprima "\nnumero do registro: ", reg;
    imprima "\nvalor do imposto: R$ ", formatar(imposto,2);
    imprima "\nnumero de meses em atraso: ", meses;
    imprima "\nvalor da multa: R$ ", formatar(multa,2);
    imprima "\nvalor total: R$ ", formatar(imposto + multa,2);
    imprima "\n\n";
    imprima "\ndigite numero do registro do imovel ou 0 para terminar: ";
    leia reg;
  }
  imprima "\n";
fimprog
```

algoritmo 338

A associação dos fazendeiros de gado gostaria de fazer um algoritmo que pudesse entrar com o código de cada fazenda (int), código da manada (int), tipo da manada (1 – vaca, 2 – touro, 3 – boi, 4 – bezerro) e quantidade de cabeças por manada (int). Sabendo-se que cada fazenda pode possuir mais de uma manada e o algoritmo se encerra quando se digita -1 para o código da fazenda, imprima:

- *código de cada fazenda;*
- *para cada tipo de manada da fazenda, total de cabeças;*
- *número total de manadas da fazenda;*
- *número total de fazendas;*
- *para cada tipo de manada da associação, total de cabeças;*
- *número total de manadas da associação ;*
- *quantidade total de cabeças de gado prontas para o corte.*

```
prog enq77
 int cfaz, tipo, qcm, q1faz, q2faz, q3faz, q4faz, q1afa, q2afa, q3afa;
 int q4afa, ntmfaz, ntmafa, qcfaz, qcafa, ntfafa ;
 q1afa <- 0;
 q2afa <- 0;
 q3afa <- 0;
 q4afa <- 0;
 qcafa <- 0;
 ntfafa <- 0;
 ntmafa <- 0;
 imprima "\ncodigo da fazenda ou -1 para terminar: ";
 leia cfaz;
 enquanto( cfaz <> -1 )
 { ntfafa++;
   ntmfaz <- 0;
   q1faz <- 0;
   q2faz <- 0;
```

```
  q3faz <- 0;
  q4faz <- 0;
  qcfaz <- 0;
  imprima "\ndigite tipo de manada 1 vaca  2 touro 3  boi  4 bezerro ou
  5 para terminar: ";
  leia tipo;
  enquanto(tipo < > 5)
  {  ntmfaz++;
     imprima "\ndigite quantidade de cabecas: ";
     leia qcm;
     se(tipo ==1 )
      { q1faz <- q1faz + qcm; }
      senao
      {  se(tipo ==2 )
         { q2faz <- q2faz + qcm; }
         senao
         { se(tipo == 3 )
           { q3faz <- q3faz + qcm; }
           senao
           { se(tipo == 4 )
              { q4faz <- q4faz + qcm; }
            }
         }
      }
     imprima "\ndigite tipo de manada 1 vaca  2 touro 3  boi  4 bezerro
     ou 5 para terminar: ";
     leia tipo;
   }
   imprima "\ndigite quantidade de cabecas para corte: ";
   leia qcfaz;
   qcafa <- qcafa + qcfaz;
   ntmafa <- ntmafa + ntmfaz;
   q1afa <- q1afa + q1faz;
   q2afa <- q2afa + q2faz;
   q3afa <- q3afa + q3faz;
   q4afa <- q4afa + q4faz;
   imprima "\ncodigo da fazenda: ", cfaz;
   imprima "\nquantidade de cabecas do tipo 1: ", q1faz;
   imprima "\nquantidade de cabecas do tipo 2: ", q2faz;
   imprima "\nquantidade de cabecas do tipo 3: ", q3faz;
   imprima "\nquantidade de cabecas do tipo 4: ", q4faz;
   imprima "\nquantidade de manadas: ", ntmfaz;
   imprima "\ncodigo da fazenda ou -1 para terminar: ";
   leia cfaz;
}
imprima "\nnumero total de fazendas: ", ntfafa;
imprima "\nquantidade de cabecas do tipo 1: ", q1afa;
imprima "\nquantidade de cabecas do tipo 2: ", q2afa;
imprima "\nquantidade de cabecas do tipo 3: ", q3afa;
```

```
 imprima "\nquantidade de cabecas do tipo 4: ", q4afa;
 imprima "\nquantidade de manadas: ", ntmafa;
 imprima "\nquantidade de cabecas para corte: ", qcafa;
 imprima "\n";
fimprog
```

algoritmo 339

Segundo Goldbach, qualquer número par pode ser o resultado da soma de dois números primos. Criar um algoritmo que possa entrar com vários números enquanto forem pares e, para cada número, imprimir todos os arranjos (a ordem importa) possíveis entre dois números primos cuja soma seja igual ao número.

```
prog enq78
 int num,p1,p2,c,c1, c2;
 imprima "\ndigite um numero maior do que 0: ";
 leia num;
 enquanto( num % 2==0 && num > 2)
 {   p1<-2;p2<-2;
    enquanto(p1<=num-p2)
    { c1 <- 0; c <- 2;
      enquanto( c1 == 0 && c <= p1 div 2 )
       { se( p1 % c ==0)
         { c1<-1; }
         c++;
       }
       se( c1==0 )
       { p2<-2;
         enquanto( p2 <= num-p1 )
         {
          c2<-0; c<-2;
          enquanto( c2 == 0 && c <= p2 div 2 )
          { se( p2 % c ==0)
            { c2<-1; }
            c++;
          }
          se( c2==0 && p1+p2 == num )
          {  imprima "\n", p1, " + ", p2," = ", p1 + p2;}
          p2++;
         }
      }
      p1++;
      p2 <- 2;
   }
   imprima "\ndigite um numero maior do que 0: ";
   leia num;
 }
 imprima "\n";
fimprog
```

algoritmo 340

Criar um algoritmo que entre com numerador e denominador de duas frações e imprima:

- *a soma algébrica das frações;*
- *se for possível, representar a fração sob a forma de número misto.*

Observações:
1. para somar duas frações, os denominadores precisam ser iguais e, se não forem, reduzir ao mesmo denominador e somar;
2. para representar sob a forma de número misto, só se a fração for imprópria, isto é, numerador >= denominador.

```
prog enq79
 int n1, n2,nf, d1, d2,d, D,num1, num2,pinteira, pfrac, numeradorfracao;
 int denominadorfracao, soma;
 imprima "\nEntre com 1 numerador: ";
 leia n1;
 imprima "\nEntre com 1 denominador: ";
 leia d1;
 enquanto( d1 <= 0 )
 { imprima "\nDENOMINADOR INVALIDO";
   imprima "\nEntre com 1 denominador: ";
   leia d1;
 }
 imprima "\nEntre com 2 numerador: ";
 leia n2;
 imprima "\nEntre com 2 denominador: ";
 leia d2;
 enquanto( d2 <= 0 )
 { imprima "\nDENOMINADOR INVALIDO";
   imprima "\nEntre com 2 denominador: ";
   leia d2;
 }
 se( d1 == d2 )
 { numeradorfracao <- n1 + n2; denominadorfracao <- d1; soma <- d1;}
 senao
 { se( d1 < d2 )
   { soma <- d2;d <- d1;D<-d2;}
   senao
   { soma <- d1;d<-d2; D<-d1;}
   enquanto( soma % d <> 0 )
   { soma <-soma + D;}
     num1<- (soma div d1) * n1;
     num2 <- ( soma div d2) * n2;
     numeradorfracao<-num1 + num2; denominadorfracao <- soma;
 }
```

```
 imprima "\na soma das fracoes: ",n1," / ", d1," e ", n2," / ",d2,"  = ",
           numeradorfracao," / ", denominadorfracao; # continuacao
 nf <- abs(numeradorfracao); pinteira <- nf div denominadorfracao;
 se( numeradorfracao < 0 )
 { pinteira <- pinteira *(-1); }
 se( nf % denominadorfracao == 0 )
 { imprima "\n\nnumero inteiro == ",pinteira;}
 senao
 {se( nf > denominadorfracao )
 { pfrac <- nf % denominadorfracao;
  imprima "\n\nnumero misto =", pinteira, "  ", pfrac," /  ",  soma;}
  senao
  {imprima "\n\nfracao propria";} }
 imprima "\n";
fimprog
```

algoritmo 341

Criar um algoritmo que funcione através do menu a seguir:

> *MAQUINA ESPERTA*
>
> *1 – Soma varios numeros*
> *2 – Multiplica varios numeros*
> *3 – Sai do algoritmo*
> *OPCAO*

```
prog enq80
 int op, n;
 real num, soma, prod;
 faca
 {
   imprima "\nMAQUINA ESPERTA ";
   imprima "\n1 - Soma varios numeros ";
   imprima "\n2 - Multiplica varios numeros ";
   imprima "\n3 - Sai do algoritmo ";
   imprima "\nOPCAO: ";
   leia op;
   se( op == 1 )
   { soma <- 0.;
     imprima "\ndigite numero ou -0.0001 para finalizar:";
     leia num;
     enquanto( num <> -0.0001 )
     { soma <- soma + num;
       imprima "\ndigite numero ou -0.0001 para finalizar:";
       leia num;
     }
     imprima "\nsoma: " , soma;
```

```
  }
  senao
  { se( op == 2 )
    { prod <- 1.;
      imprima "\ndigite numero ou -0.0001  para finalizar: ";
      leia num;
      enquanto( num <> -0.0001 )
      { prod <- prod * num;
        imprima "\ndigite numero ou -0.0001  para finalizar: ";
        leia num;
      }
      imprima "\nproduto: " , prod;
  }
  senao
  { se( op == 3 )
    { imprima "\nSai do algoritmo";}
    senao
    { imprima "\nOpcao nao disponivel" ;}
  }
 }
  imprima "\n\n";
 }
 enquanto( op <> 3)
 imprima "\n";
fimprog

prog enq80
 # nao funciona na versao 2 do UAL
 int op, n;
 real num, soma, prod;
 faca
 {
  imprima "\nMAQUINA ESPERTA ";
  imprima "\n1 - Soma varios numeros ";
  imprima "\n2 - Multiplica varios numeros ";
  imprima "\n3 - Sai do algoritmo ";
  imprima "\nOPCAO: ";
  leia  op;
  escolha(op)
  {
  caso 1:
   soma <- 0.;
   imprima "\ndigite numero ou -0.0001  para finalizar:";
   leia num;
   enquanto( num  <> -0.0001 )
   { soma <- soma + num;
     imprima "\ndigite numero ou -0.0001  para finalizar:";
     leia num;
```

```
        {  imprima "\nNao faco raiz de numero negativo: ";}
      }
      senao
      {
       se(  op == "F" || op == "f"  )
       {  imprima "\nSaindo: "; }
       senao
       { imprima "\nOpcao nao disponivel"; }
 }}}
 imprima "\n\n\n\n";
 }
 enquanto(op < >"F" && op< >"f"  )
 imprima "\n";
fimprog
```

algoritmo 343

Criar um algoritmo que funcione através do menu a seguir

> *MENU*
>
> *A – Armazena na variável menor e imprime o nome que tiver o menor número de caracteres entre três nomes*
> *B – Brinca com a palavra*
> *C – Calcula e imprime a tangente de um ângulo em graus*
> *F – Termina o algoritmo*
> *OPÇÃO:*

Observações:

1. *Na entrada de dados, considerar as letras maiúsculas e minúsculas.*
2. *No item B, se você entrar com a palavra SORTE, deverá sair:*

   SORTE

   SORT

   SOR

   SO

   S

3. *Não se esqueça de testar os ângulos que não têm tangenteS.*

```
prog enq82
 string nome, menor, palavra, op;
 int  x, l, i, cont;
 real ang, rang;
 faca
 {
  imprima "\nMENU";
```

```
imprima "\nA - Armazena na variavel menor e imprime o nome que tiver
o menor numero de caracteres entre 3 nomes ";
imprima "\nB - Brinca com a palavra";
imprima "\nC - Calcula e imprime a tangente de um angulo em graus";
imprima "\nF - Termina o algoritmo";
imprima "\nOPCAO: ";
leia op;
se(  op == "A" || op == "a"  )
{
  imprima "\n Digite um nome: ";
  leia nome;
  menor <- nome;
  para( x  <- 1 ; x <= 2;  x++)
  {
   imprima "\n Digite um nome: ";
   leia  nome;
   se(  strtam(nome) < strtam(menor) )
   {  menor <- nome; }
   }
   imprima "\nNome com menor numero de caracteres: ", menor;
  }
  senao
  {
   se(  op == "B" || op == "b"  )
   {
     imprima "\npalavra: ";
     leia  palavra ;
     para( i<- strtam(palavra); i>=1; i--)
     {
     imprima "\n", strnprim(palavra,i);
     }
    }
    senao
    {
      se(  op == "C" || op == "c"  )
      {
        imprima "\nangulo: ";
        leia ang;
        rang <-ang * pi / 180;
        se(cos(rang)< > 0.)
        { imprima "\n tangente: ", tan(rang) ; }
        senao
        {  imprima "\n Tangente tende a infinito: ";}
       }
       senao
       {
        se(  op == "F" || op == "f"  )
        {  imprima "\nSaindo: "; }
```

```
          senao
          { imprima "\nOpcao nao disponivel"; }
  }}}
 imprima "\n\n\n\n";
 }
 enquanto(op< >"F" && op< >"f"  )
 imprima "\n";
fimprog
```

algoritmo 344

Criar um algoritmo que funcione através do menu a seguir.

> *MENU*
>
> *A – Entra com uma frase e armazena em outra variável a frase invertida e imprime*
> *B – Entra com um número e seu número de dígitos e imprime invertido*
> *C – Calcula e imprime a tangente de um ângulo em graus*
> *F – Termina o algoritmo*
> *OPCAO:*

Observações:

1. *Na entrada de dados, considerar as letras maiúsculas e minúsculas.*
2. *No item B, se você entrar com 987654 e 6, deverá sair: 456789*
3. *No item C, não se esqueça de testar os ângulos que não têm tangente.*

```
prog enq83
 string nome, menor, frase, frasei, op, letra;
 int  x, i, cont, num, numi, c, n, numd;
 real ang, rang;
 faca
 {
  imprima "\nMENU";
  imprima "\nA - Entra com uma frase e armazena em outra variavel a frase
           invertida e imprime"; # continuacao
  imprima "\nB - Entra com um numero e seu numero de digitos e imprime
           invertido"; # continuacao
  imprima "\nC - Calcula e imprime a tangente de um angulo em graus";
  imprima "\nF - Termina o algoritmo";
  imprima "\nOPCAO: ";
  leia  op;
   se(  op == "A" || op == "a"  )
   {
     imprima "\n Digite um frase: ";
     leia frase;
     frasei <-"";
```

```
    cont <- strtam(frase) - 1;
    para(  x <- cont; x>= 0 ; x-)
    { letra <- strelem( frase, x);
    frasei <- strconcat(frasei,letra );
    }
    imprima "\nfrase invertida: ", frasei;
  }
  senao
  {
    se(  op == "B" || op == "b"  )
    {
      imprima "\nnumero: ";
      leia num;
      imprima "\nnumero de digitos: ";
      leia numd;
      numi <- 0;
      n <- num;
      para( c<- 1; c <= numd; c++)
      {
         numi<- numi * 10 + n % 10;
         n <- n  div  10;
      }
      imprima "\n numero invertido: ",numi;
    }
    senao
    {
     se(  op == "C" || op == "c"  )
     {
      imprima "\nangulo: ";
      leia ang;
      rang <-ang * pi / 180;
      se(cos(rang)< > 0.)
      {
       imprima "\nTangente: ", tan(rang) ; }
      senao
      {  imprima "\nTangente tende a infinito: ";}
    }
    senao
    {
      se(  op == "F" || op == "f"  )
      {  imprima "\nSaindo: "; }
      senao
      {imprima "\nOpcao nao disponivel"; }
 }}}
 imprima "\n\n\n\n";
 }
 enquanto(op < > "F" && op< >"f"  )
 imprima "\n";
fimprog
```

algoritmo 345

Criar um algoritmo que funcione através do menu a seguir.

> *MENU*
> *1 – Lê dez palavras e exibe a menor delas.*
> *2 – Lê uma palavra e depois armazena a letra W em todas as posições pares da palavra.*
> *3 – Lê uma frase e exibe o número de palavras existentes na frase*
> *4 – Sai do algoritmo*
> *Opção:*

```
prog enq84
 int op, cont, aux;
 string nome, frase, palavra;
 faca
 {
  imprima "\nMenu";
  imprima "\n1 - Le dez palavras e exibe a menor delas";
  imprima "\n2 - Le uma palavra e imprima a letra W em todas as posicoes
            pares da palavra."; # continuacao
  imprima "\n3 - Le uma frase e exibe o numero de palavras existentes na
            frase"; # continuacao
  imprima "\n4 - Sai do algoritmo";
  imprima "\nDigite sua opcao: ";
  leia op;
  se( op == 1)
  { imprima "\nDigite a palavra: ";
    leia  palavra ;
    nome <- palavra;
    para(cont <- 1; cont <= 9; cont++)
    { imprima "\nDigite a palavra: ";
      leia  palavra ;
      se( strtam( palavra) < strtam( nome))
      {  nome <- palavra; }
    }
    imprima "\nA menor palavra digitada foi: ", nome;
  }
  senao
  { se( op == 2)
    { imprima "\nDigite uma palavra";
      leia nome;
      para( cont <- 0; cont <= strtam(nome) - 1; cont++)
      { se(cont % 2 == 0)
        {  imprima "W";  }
```

```
        senao
        { imprima strelem(nome, cont);}
      }
     imprima "\n";
    }
    senao
    { se( op == 3)
      { imprima "\nDigite uma frase: ";
        leia frase;
        aux <- 1;
        para( cont <- 0; cont <= strtam(frase) - 1; cont++)
        { se(strelem(frase, cont) == " ")
          { aux++;  }
       }
       imprima "\nA frase lida possui ", aux, " palavras";
      }
      senao
      { se(  op ==4)
        { imprima "\nEncerrando algoritmo"; }
        senao
        { imprima "\nOpcao invalida";}
       }
      }
     }
     imprima "\n\n";
 }
 enquanto(  op < > 4 )
 imprima "\n";
fimprog
```

algoritmo 346

Criar um algoritmo que funcione através do menu a seguir:

> *MENU*
>
> *1 – Imprime o comprimento da frase*
> *2 – Imprime os dois primeiros e os dois últimos caracteres da frase*
> *3 – Imprime a frase espelhada*
> *4 – Termina o algoritmo*
> *OPCAO*

```
prog enq85
 string frase;
 int op, x, tam;
 faca
 {
```

```
   imprima "\nMENU";
   imprima  "\n1 - Imprime o comprimento da frase ";
   imprima  "\n2 - Imprime os dois primeiros e os dois ultimos caracteres
   da frase";
   imprima  "\n3 - Imprime a  frase espelhada ";
   imprima  "\n4 - Termina o algoritmo ";
   imprima "\nOPCAO: ";
   leia  op;
   se(op == 1)
   { imprima "\nDigite uma frase: ";
     leia  frase ;
     imprima "\nnumero de caracteres da frase: ", strtam(frase);}
   senao
   { se(op == 2)
     { imprima "\nDigite uma frase: ";
       leia  frase ;
       imprima "\nos dois primeiros caracteres: ", strnprim(frase,2);
       tam <- strtam(frase) -2;
       imprima "\nos dois ultimos caracteres: ", strnresto(frase, tam);}
     senao
     { se(op == 3)
       { imprima "\nDigite uma frase: ";
         leia  frase ;
         para(x <-strtam(frase) -1; x >=0; x–)
         { imprima strelem(frase,x);}
       }
       senao
       { se(op == 4)
         { imprima "\nFim do algoritmo"; }
         senao
         { imprima "\nopcao nao disponivel";}
       }
     }
   }
   imprima "\n\n";
  }
  enquanto(op < > 4)
  imprima "\n\n";
fimprog

prog enq85
 # nao funciona na versao 2 do UAL
 string frase;
 int op, x, tam;
 faca
 {
  imprima "\nMENU";
  imprima  "\n1 - Imprime o comprimento da frase ";
```

```
 imprima  "\n2 - Imprime os dois primeiros e os dois ultimos caracteres
 da frase";
 imprima  "\n3 - Imprime a  frase espelhada ";
 imprima  "\n4 - Termina o algoritmo ";
 imprima "\nOPCAO: ";
 leia  op;
 escolha(op)
 {
  caso 1:
   imprima "\nDigite uma frase: ";
   leia  frase ;
   imprima "\nnumero de caracteres da frase: ", strtam(frase);
   pare;
  caso 2:
   imprima "\nDigite uma frase: ";
   leia  frase ;
   imprima "\nos dois primeiros caracteres: ", strnprim(frase,2);
   tam <- strtam(frase) -2;
   imprima "\nos dois ultimos caracteres: ", strnresto(frase, tam);
   pare;
  caso 3:
   imprima "\nDigite uma frase: ";
   leia  frase ;
   para(x <-strtam(frase) -1; x >=0; x--)
   { imprima strelem(frase,x);
   pare;
  caso 4:
   imprima "\nFim do algoritmo";
   pare;
  senao
   imprima "\nopcao nao disponivel";
 }
 imprima "\n\n";
 }
 enquanto(op <> 4)
 imprima "\n\n";
fimprog
```

Capítulo 5

Estruturas homogêneas: vetores e matrizes

CONCEITO GERAIS

Apesar de você ter feito uso das estruturas de repetição que lhe permitiram a entrada de vários dados, ainda não foi possível o armazenamento de todos esses dados, de uma vez só, ou não, pois lhe faltava conhecer a estrutura do vetor.

Um vetor é um arranjo de elementos armazenados na **MP**, um após o outro, todos com o mesmo nome. A idéia é a mesma de uma **matriz linha** da matemática, isto é, várias colunas e uma linha.

Assumiremos que a primeira posição do vetor é 0.

2	4	5	8	12	3	56	34
0	1	2	3	4	5	6	7

A [2   4   5   8   12   3   56   34]

Esse é um vetor de 8 elementos, isto é, tem 8 variáveis, todas com o mesmo nome e diferentes por sua posição dentro do arranjo que é indicada por um **índice**.

Quando se tem somente uma linha, podemos omiti-la e colocar somente a coluna.

$A_0 = 2$ $A_1 = 4$ $A_2 = 5$ $A_3 = 8$ $A_4 = 12$ $A_5 = 3$ $A_6 = 56$ $A_7 = 34$

Em algoritmos, representamos da seguinte forma:

A[0] = 2 A[1] = 4 A[2] = 5 A[3] = 8 A[4]= 12 A[5] = 3 A[6] = 56 A[7] = 34

Para dimensionar um vetor, usamos o seguinte comando na declaração de variáveis:

```
tipo nome[dimensão]
```

onde **dimensão,** na prática, é o número de elementos:

[5] 5 elementos (de 0 a 4)

tipo poderá ser: **int, real** ou **string**
nome será o que você dará ao vetor dentro das regras para nomear uma variável

No exemplo acima seria:

int A[8]

Em algoritmos também podemos ter vetor caracter e **lembre-se: a variável caracter é armazenada na MP como sendo um vetor, mesmo sem a declararmos como tal.** Para melhor entendimento, a representação será na vertical.

Linguagens que ASSUMEM a posição

real A[5];
int B[10];
string NOMES[20];

A

0	
1	
2	
3	
4	

B

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

NOMES

	0	1	...			
0						
1						
2						
3						
4						
5						
6						
...						
18						
19						

Um vetor string, na verdade, é uma matriz, pois como já falamos cada variável string é um vetor.

Um algoritmo com vetor implica vários trechos para que possa funcionar corretamente. Esses trechos são **independentes.**

- TRECHO DE DIMENSIONAMENTO – já visto anteriormente
- TRECHO DE ENTRADA DE DADOS
  - normalmente, uma **estrutura de repetição**
  - se for a estrutura do **para,** deverá ter o **valor final** igual à **última posição** do vetor.
  - se for a estrutura do **enquanto** ou **faca enquanto,** deverá ter uma variável que será incrementada e **nunca** poderá assumir um valor maior do que a **última posição** do vetor.

```
para( L<- 0; L < tamanho do vetor ; L++)
{
 imprima ... ;
 leia  nomedovetor [ L ];
}
```

  - o trecho anterior poderá ser incrementado com outros comandos;
  - **nunca deverá ter um comando *imprima* tendo como argumento o nomedovetor [L];**
  - poderá ter: estrutura de **seleção, atribuição,** outro **para etc.**

- TRECHO DE SAÍDA
  - normalmente, uma **estrutura de repetição;**
  - se for a estrutura do **para,** deverá ter o **valor final** igual à **última posição** do vetor;
  - se for a estrutura do **enquanto** ou **faca enquanto,** deverá ter uma variável que será incrementada e **nunca** poderá assumir um valor maior do que a **última posição** do vetor.

```
para( L<- 0; L < tamanho do vetor; L++)
{
  imprima "\n", nomedovetor [ L ] ;
}
```

  - o trecho acima poderá ser incrementado de outros comandos;
  - **nunca deverá ter um comando *leia* tendo como argumento o nomedovetor [L];**
  - poderá ter: estrutura de **seleção, atribuição,** outro **para etc.**

algoritmo 347

Criar um algoritmo que entre com dez nomes e imprima uma listagem contendo todos os nomes.

```
prog lerimp
 int L ;
 string nomes[10];
 # trecho de entrada
 para( L<- 0; L <= 9 ; L++)
 {
   imprima "\ndigite ", L + 1," nome: ";
   leia nomes [L] ;
 }
 # fim do trecho de entrada
 # trecho de saida
 imprima "\n\n";
 para( L<- 0; L <= 9 ; L++)
 {
  imprima "\n",  nomes[ L];
 }
 # fim do trecho de saida
 imprima "\n";
fimprog
```

digite 1 nome: ANITA
digite 2 nome: JOAO
digite 3 nome: PEDRO
digite 4 nome: MARIA
digite 5 nome: FILIPE
digite 6 nome: ANGELA
digite 7 nome: ANA
digite 8 nome: REGINA
digite 9 nome: LUIZ
digite 10 nome: TEREZA

ANITA
JOAO
PEDRO
MARIA
FILIPE
ANGELA
ANA
REGINA
LUIZ
TEREZA

Quando se têm dados para mais de um vetor relativos a uma pessoa ou a um objeto, os trechos de entrada são separados? **NÃO devem ser (exemplo: prog** imprimemedia).

Quando se têm dados para mais de um vetor que não têm nada em comum, os trechos de entrada são separados? **DEVEM ser** (exemplo: **prog** somavetores).

algoritmo 348

Criar um algoritmo que armazene nome e duas notas de 5 alunos e imprima uma listagem contendo nome, as duas notas e a média de cada aluno.

```
prog imprimemedia
 int L;
 string nomes[5];
 real pr1[5], pr2[5], media[5];
```

```
# trecho de entrada
para( L<- 0; L <= 4 ; L++)
{
  imprima "\ndigite ", L + 1, " nome: ";
  leia nomes[ L ] ;
  imprima "digite 1ª nota: ";
  leia pr1[L];
  imprima "digite 2ª nota: ";
  leia pr2[L];
  media[L] <- (pr1[L] + pr2[L])/2;
}
# fim do trecho de entrada
#trecho de saida
imprima "\n\n\n\t\t\t\tRELACAO FINAL\n";
para( L<- 0; L <= 4 ; L++)
{ imprima "\n", L+1, "- ", nomes[L];
  imprima "\n",pr1[L],"\t", pr2[L], "\t",media[L];}
# fim do trecho de saida
imprima "\n";
fimprog
```

algoritmo 349

Criar um algoritmo que armazene números em dois vetores inteiros de cinco elementos cada. Gere e imprima o vetor soma.

```
prog somavetores
 int L, a[5], b[5], soma[5];
 para( L<- 0; L <= 4 ; L++)
 {
  imprima "\nelemento do vetor
  A[", L + 1, "]: ";
  leia  a[ L];
 }
 imprima "\n";
 para( L<- 0; L <= 4 ; L++)
 {
 imprima "\nelemento do vetor
  B[", L + 1, "]: ";
 leia  b[ L];
 soma[L] <- a[L] + b[L];
 }
 imprima "\n\n\nVetor Soma\n";
 para( L<- 0; L <= 4 ; L++)
 { imprima "\nsoma[",L + 1, "] -
", soma[ L];}
 imprima "\n";
fimprog
```

elemento do vetor A[1]: 12
elemento do vetor A[2]: 10
elemento do vetor A[3]: 9
elemento do vetor A[4]: 4
elemento do vetor A[5]: 5

elemento do vetor B[1]: 8
elemento do vetor B[2]: 10
elemento do vetor B[3]: 11
elemento do vetor B[4]: 16
elemento do vetor B[5]: 15

Vetor Soma

soma[1] - 20
soma[2] - 20
soma[3] - 20
soma[4] - 20
soma[5] - 20

ORDENANDO VETORES

Compare o elemento que está na 1ª posição com todos os seguintes a ele. Quando for necessário, troque-o de lugar, fazendo uso da variável **aux**. Repita essa operação até que tenha feito todas as comparações.

1ª Fase

1º passo: Se o nome da posição **0** na ordem alfabética vier depois do nome na posição 1, então inverta-os. Como isso não acontece, tudo fica igual.

NOMES

0	*JOAO*
1	*RUI*
2	*IVO*
3	*BIA*
4	*ANA*

AUX

NOMES

0	*JOAO*
1	*RUI*
2	*IVO*
3	*BIA*
4	*ANA*

2º passo: Se o nome da posição 0 na ordem alfabética vier depois do nome na posição 2, então inverta-os.

NOMES

0	*JOAO*
1	*RUI*
2	*IVO*
3	*BIA*
4	*ANA*

AUX

JOAO

NOMES

0	*IVO*
1	*RUI*
2	*JOAO*
3	*BIA*
4	*ANA*

3º passo: Se o nome da posição **0** na ordem alfabética vier depois do nome na posição 3, então inverta-os.

NOMES

0	*IVO*
1	*RUI*
2	*JOAO*
3	*BIA*
4	*ANA*

AUX

IVO

NOMES

0	*BIA*
1	*RUI*
2	*JOAO*
3	*IVO*
4	*ANA*

4º passo: Se o nome da posição **0** na ordem alfabética vier depois do nome na posição **4**, então inverta-os.

	NOMES
0	BIA
1	RUI
2	JOAO
3	IVO
4	ANA

AUX
BIA

	NOMES
0	ANA
1	RUI
2	JOAO
3	IVO
4	BIA

- Após essas comparações e trocas, conseguimos colocar na posição 0 o nome ANA.
- Partiremos para a busca do segundo nome da lista.
- Não mais nos preocuparemos com a posição 0.

2ª Fase

1º passo: Se o nome da posição **1** na ordem alfabética vier depois do nome na posição **2**, então inverta-os.

	NOMES
0	ANA
1	RUI
2	JOAO
3	IVO
4	BIA

AUX
RUI

	NOMES
0	ANA
1	JOAO
2	RUI
3	IVO
4	BIA

2º passo: Se o nome da posição **1** na ordem alfabética vier depois do nome na posição **3**, então inverta-os.

	NOMES
0	ANA
1	JOAO
2	RUI
3	IVO
4	BIA

AUX
JOAO

	NOMES
0	ANA
1	IVO
2	RUI
3	JOAO
4	BIA

3º passo: Se o nome da posição **0** na ordem alfabética vier depois do nome na posição **4**, então inverta-os.

	NOMES
0	*ANA*
1	*IVO*
2	*RUI*
3	*JOAO*
4	*BIA*

AUX
IVO

	NOMES
0	*ANA*
1	*BIA*
2	*RUI*
3	*JOAO*
4	*IVO*

- Após essas comparações e trocas, conseguimos colocar na posição 1 o nome BIA.
- Partiremos para a busca do terceiro nome da lista.
- Não mais nos preocuparemos com as posições 0 e 1.

3ª Fase

1º passo: Se o nome da posição **2** na ordem alfabética vier depois do nome na posição **3**, então inverta-os.

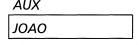
	NOMES
0	*ANA*
1	*BIA*
2	*RUI*
3	*JOAO*
4	*IVO*

AUX
RUI

	NOMES
0	*ANA*
1	*BIA*
2	*JOAO*
3	*RUI*
4	*IVO*

2º passo: Se o nome da posição **2** na ordem alfabética vier depois do nome na posição **4**, então inverta-os.

	NOMES
0	*ANA*
1	*BIA*
2	*JOAO*
3	*RUI*
4	*IVO*

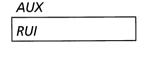
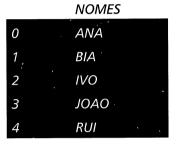
AUX

	NOMES
0	*ANA*
1	*BIA*
2	*IVO*
3	*RUI*
4	*JOAO*

- Após essas comparações e trocas, conseguimos colocar na posição 1 o nome BIA.
- Partiremos para a busca do terceiro nome da lista.
- Não mais nos preocuparemos com as posições 0, 1 e 2.

4ª Fase

1º passo: Se o nome da posição 3 na ordem alfabética vier depois do nome na posição 4, então inverta-os.

	NOMES
0	*ANA*
1	*BIA*
2	*IVO*
3	*RUI*
4	*JOAO*

	NOMES
0	*ANA*
1	*BIA*
2	*IVO*
3	*JOAO*
4	*RUI*

Conclusões

- Apesar de termos cinco nomes, só tivemos quatro fases, pois quando comparamos o penúltimo com o último, economizamos uma fase.
- O número de comparações foi diminuindo em cada fase. Isso se explica porque o número de combinações também estava diminuindo, uma vez que diminuía o número de nomes abaixo do que estava em evidência.
- Vamos tentar montar o trecho de ordenação:

1ª linha:

```
para( L <-0;  L < 4 ;  L++)
```

Por que de **0** até **<** 4?
Porque as fases envolveram as buscas das posições **0**,**1**, **2** e **3**.

2ª linha:

```
{
```

Inicia o bloco do para

3ª linha:

```
para( c <- L + 1;  c <= 4 ;  c ++)
```

Por que de L **+** 1 até **<=** 4?
Porque, em cada fase, comparávamos a posição em evidência com todas as outras posições até a última.

4ª linha:

```
{
```

Inicia o bloco do para

5ª linha:

```
se( nomes[L] >  nomes[c] )
```

Pense no **L** como sendo a **fase** e, em **c** como sendo uma das **etapas da fase.**

6ª linha:

```
{
```

Inicia o bloco do se

7ª linha/8ª linha/9ª linha:

```
aux <- nomes[L] ;
nomes[L] <- nomes[c] ;
nomes[c] <- aux;
```

Este trecho permite a troca entre as duas variáveis.

10ª linha:

```
}
```

Finaliza o bloco do se

11ª linha:

```
}
```

Finaliza o bloco do para

12ª linha:

```
}
```

Finaliza o bloco do para

Trecho de ordenação

```
para( L <-0;  L < 4 ;  L++)
{
  para( c <- L + 1;  c <= 4 ;  c ++)
  {
   se( nomes[L] >  nomes[c] )
   {
    aux <- nomes[L] ;
    nomes[L] <- nomes[c] ;
    nomes[c] <- aux;
   }
  }
}
```

↳ *Se precisarmos ordenar um vetor e se tivermos outros vetores dependentes dele, precisaremos fazer vários trechos de troca. Isto é, todo trecho que se encontra no retângulo mais interno.*

Alguns algoritmos básicos:

algoritmo 350

Criar um algoritmo que armazene 5 nomes em um vetor. Ordenar e imprimir uma listagem.

```
prog ordenanomemevetor
 int L, c;
 string nomes[5], aux;
 para( L<- 0; L <= 4 ; L++)
 {  imprima "\nnome: " ;  leia nomes[L]
 ;}
 para( L<- 0; L <= 3 ; L++)
 {   para( c<- L + 1; c <= 4 ; c++)
    {  se(  nomes[L] > nomes[c] )
        {  aux <- nomes[L];
           nomes[L] <- nomes[c ];
           nomes[c] <- aux;    }
     } }
 imprima "\n\n " ;
 para( L<- 0; L <= 4 ; L++)
 {   imprima "\n", L +1, " - ", nomes[L]
 ;}
 imprima "\n";
fimprog
```

nome: GUTO GARCIA
nome: PEDRO CORREA
nome: ANITA LOPES
nome: MARIA CORREA
nome: JOAO BOND

ANITA LOPES
GUTO GARCIA
JOAO BOND
MARIA CORREA
PEDRO CORREA

algoritmo 351

Criar um algoritmo que armazene cinco nomes em um vetor e depois possa ser digitado um número que corresponde a uma pessoa e imprimir esse nome.

```
prog procuranomepelonumero
 int L, num ;
 string nomes [5];
 para( L<- 0; L <= 4 ; L++)
 {  imprima "\nnome " , L + 1, ": ";  leia
 nomes[L] ;}
 #entra com numero para procurar pela
 posicao
 imprima "\nDigite o numero da pessoa: ";
 leia num;
 enquanto(num < 1 ||  num > 5)
```

nome 1: ANITA
nome 2: GUTO
nome 3: JOAO
nome 4: PEDRO
nome 5: FILIPE

Digite o numero da pessoa: 3

JOAO

```
 {
 imprima "\n Numero fora do intervalo";
 imprima "\n Digite o numero da pessoa: ";
 leia num;
 }
 imprima "\n", nomes[ num -1] ;
 imprima "\n";
fimprog
```

algoritmo 352

Criar um algoritmo que armazene cinco nomes em um vetor e depois possa ser digitado um nome e, se for encontrado, imprimir a posição desse nome no vetor; caso contrário, imprimir uma mensagem.

```
prog procuranomepelonome
 # procura pelo nome
 int L ;
  string nome, nomes [5];
  para( L<- 0; L <= 4 ; L++)
  { imprima "\nnome: " ;
    leia nomes[L] ;
  }
  # entra com nome para procura
   imprima "\n Digite nome para
procura: " ;
   leia nome;
   L <-0;
   enquanto( L<4 &&
nome< >nomes[L]  )
   { L++;  }
   se(nome ==nomes[L] )
  {
   imprima "\nencontrei na
posicao: ", L+1;
  }
  senao
  {
    imprima "\nnao encontrei" ;
  }
 imprima "\n";
fimprog
```

nome: ANITA

nome: GUTO

nome: JOAO

nome: PEDRO

nome: FILIPE

Digite nome para procura: ANITA LOPES

nao encontrei

nome: ANITA

nome: GUTO GARCIA

nome: JOAO

nome: PEDRO

nome: FILIPE

Digite nome para procura: GUTO GARCIA

encontrei na posicao: 2

EXERCÍCIOS – LISTA 5
LISTA DE VETORES

algoritmo 353

Armazenar 10 nomes em um vetor NOME e imprimir uma listagem numerada.

```
prog vetor1
 int L;
 string nomes[10];
 # trecho de entrada de 10 nomes
 para( L<- 0; L <= 9 ; L++)
 {
    imprima "\nDigite ", L + 1, " nome: ";
    leia nomes[ L ] ;
 }
 # fim do trecho de entrada
 # trecho de saida
 imprima "\nRELACAO DAS PESSOAS\n";
 para( L<- 0; L <= 9 ; L++)
 {
    imprima "\n", L+1, "- ", nomes[ L];
 }
 # fim do trecho de saida
 imprima "\n";
fimprog
```

algoritmo 354

Armazenar 15 números inteiros em um vetor NUM e imprimir uma listagem numerada contendo o número e uma das mensagens: par ou ímpar.

```
prog vetor2
 int L , num[15];
 # trecho de entrada de 15 elementos
 para( L<- 0; L <= 14 ; L++)
 {
  imprima "\ndigite ", L + 1, " numero: ";
  leia num[ L ] ;
 }
 # fim do trecho de entrada
 # trecho de saida
 imprima "\nRELACAO DOS NUMEROS\n";
 para( L<- 0; L <= 14 ; L++)
 {
   imprima "\n", L+1, "- ", num[L];
   se(num[L] %2==0)
    { imprima " e' PAR";  }
    senao
     { imprima " e' IMPAR";  }
 }
```

```
 # fim do trecho de saida
 imprima "\n";
fimprog
```

algoritmo 355

Armazenar 8 números em um vetor e imprimir todos os números. Ao final, teremos o total de números múltiplos de seis digitados.

```
prog vetor3
 int L , cont, num[8];
 cont <- 0;
 # trecho de entrada
  para( L<- 0; L <= 7 ; L++)
  {
    imprima "\ndigite ", L + 1, " numero: ";
    leia num[ L ] ;
    se(num[L] %6==0)
    { cont++;}
  }
 # fim do trecho de entrada
 # trecho de saida
 imprima "\nRELACAO DOS NUMEROS\n";
 para( L<- 0; L <= 7 ; L++)
 {
   imprima "\n", L+1, "- ", num[L];
 }
 # fim do trecho de saida
 imprima "\n Total de multiplos de 6: ",cont;
 imprima "\n";
fimprog
```

algoritmo 356

Armazenar nomes e notas das PR1 e PR2 de 15 alunos. Calcular e armazenar a média arredondada. Armazenar também a situação do aluno: AP ou RP. Imprimir uma listagem contendo nome, notas, média e situação de cada aluno, tabulando.

```
prog vetor4
 int L ,c ,t, media[15];
 string nomes[15], sit[15];
 real pr1[15], pr2[15];
 # trecho de entrada de 15 nomes
 para( L<- 0; L <= 14 ; L++)
 {
    imprima "\n\nDigite ", L + 1, " nome: ";
    leia nomes[ L ] ;
    enquanto( strtam(nomes[L]) >30)
    {  imprima "\nNomes com ate 30 caracteres";
```

```
      imprima "\n\nDigite ", L + 1, " nome: ";
      leia nomes[ L ] ;
    }
  # trecho que garante todos os nomes com 30 caracteres para fazer tabulacao
  t<-30-strtam(nomes[L]);
  para(c<-1; c <= t; c++)
  {nomes[L]<-strconcat(nomes[L], " ");}
  imprima "\ndigite 1 nota: ";
  leia pr1[L];
  imprima "\ndigite 2 nota: ";
  leia pr2[L];
  media[L] <- realint((pr1[L] +pr2[L] )/2+0.0001);
  se(media[ L ] >= 5 )
  {  sit[ L ] <- "AP";   }
   senao
   {   sit[ L ] <- "RP";   }
 }
 # fim do trecho de entrada
 #trecho de saida
 imprima "\n\n\n\t\t\t\tRELACAO FINAL\n";
 para( L<- 0; L <= 14 ; L++)
 { imprima "\n", L+1, "- ",nomes[L],"\t",pr1[L], "\t", pr2[L],
"\t",media[L], "\t",sit[L]; }
 # fim do trecho de saida
 imprima "\n";
fimprog
```

algoritmo 357

Armazenar nome e salário de 20 pessoas. Calcular e armazenar o novo salário sabendo-se que o reajuste foi de 8%. Imprimir uma listagem numerada com nome e novo salário.

```
prog vetor5
 int L,c ,t;
 string nomes[20];
 real sal[20];
 # trecho de entrada de 20 nomes
  para( L<- 0; L <= 19 ; L++)
  {
    imprima "\ndigite ", L + 1, " nome: ";
    leia nomes[ L ] ;
    enquanto( strtam(nomes[L]) >30)
    {imprima "\nNomes com ate 30 caracteres";
     imprima "\n\nDigite ", L + 1, " nome: ";
     leia nomes[ L ] ;
    }
    t<-30-strtam(nomes[L]);
    para(c<-1; c <= t; c++)
```

```
    {nomes[L]<-strconcat(nomes[L], " ");}

    imprima "\ndigite salario: ";
    leia sal[ L ] ;
    sal[L] <-sal[L] * 1.08;
  }
 # fim do trecho de entrada
 # trecho de saida
   imprima "\nNOMES\t\t\t\t\t\n";
  para( L<- 0; L <= 19 ; L++)
   {  imprima "\n", L+1, "- ", nomes[ L],"\t",sal[L];    }
 # fim do trecho de saida
 imprima "\n";
fimprog
```

algoritmo 358

Criar um algoritmo que leia o preço de compra e o preço de venda de 100 mercadorias. O algoritmo deverá imprimir quantas mercadorias proporcionam:

- *lucro < 10 %*
- *10 % <= lucro <= 20 %*
- *lucro > 20 %*

```
prog vetor6
 real precocompra[100],precovenda[100],lucro;
 int totlucromenor10,totlucromenor20,totlucromaior20, A;
 totlucromenor10 <-  0;
 totlucromenor20 <-  0;
 totlucromaior20 <-  0;
 para(A <- 0; A < 100; A++)
 { imprima "\nPreco de compra: ";  leia precocompra [A];
   imprima "\nPreco de venda: ";  leia precovenda [A];
 }
 para(A<-0;A < 100;A++)
 {
  lucro <- precovenda[A] - precocompra[A] ;
  se(lucro<10.0)
  {totlucromenor10++;  }
  senao
  {  se(lucro <= 20.0)
     {  totlucromenor20++;  }
     senao
     { totlucromaior20++;  }
  }
 }
 imprima "\ntotal de mercadorias com lucro < 10%: ",totlucromenor10;
 imprima "\ntotal de mercadorias com 10% <= lucro <= 20%:
 ",totlucromenor20;
 imprima "\ntotal de mercadorias com lucro > 20%: ",totlucromaior20;
```

```
 imprima "\n";
fimprog
```

algoritmo 359

Criar o algoritmo que deixe entrar com nome e idade de 20 pessoas e armazene em um vetor todos os nomes que comecem pela letra do intervalo L - S.

```
prog vetor7
 string nome[20], LS[20];
 int i, c;
 c <-0;
 para(i <-0; i <= 19; i++)
 { imprima "\ndigite ", i + 1, " nome: ";
  leia nome[i];
  se(strprim(nome[i] )>= "L" &&  strprim(nome[i] )<= "S" ||
  strprim(nome[i] )>= "l" && strprim(nome[i])<= "s" )
  { LS[c] <- nome[i]; c++;}
 }
 se(c < > 0)
 { imprima "\n\nRelacao dos nomes que comecam por letra no intervalo L – S\n";
  para(i <-0; i <= c -1; i++)
 { imprima "\n",LS[i];}
 }
 senao
 {  imprima "\n\nNenhum nome encontrado";}
 imprima "\n";
fimprog
```

algoritmo 360

Criar um algoritmo que imprima o horóscopo de várias pessoas, a partir de sua data de nascimento (ddmm). O fim é determinado quando se digita 9999 para data; considere que a data foi digitada corretamente.

Mês	*Último dia*	*Signo*
01	*20*	*Capricórnio*
02	*19*	*Aquário*
03	*20*	*Peixes*
04	*20*	*Áries*
05	*20*	*Touro*
06	*20*	*Gêmeos*
07	*21*	*Câncer*
08	*22*	*Leão*
09	*22*	*Virgem*
10	*22*	*Libra*
11	*21*	*Escorpião*
12	*21*	*Sagitário*

```
prog vetor8
 int ultdia [ 12 ] , data, i, dia, mes;
 string signo[ 12 ] ;
 para( i<- 0; i< 12; i++)
 { imprima "\ndigite signo: ";
  leia signo[i];
  imprima "\ndigite ultimo dia: ";
     leia  ultdia [ i ] ;
 }
 imprima "\ndigite data no formato ddmm ou 9999 para terminar: ";
 leia data;
 enquanto( data < > 9999)
 { dia <- data div 100;
   mes <- data % 100;
   mes--;
   se( dia > ultdia [ mes ] )
   { mes <- (mes + 1 ) % 12; }
   imprima "\nsigno: ",signo[ mes ], "\n" ;
   imprima "\ndigite data no formato ddmm ou 9999 para terminar: ";
   leia data;
 }
 imprima "\n";
fimprog
```

algoritmo 361

Armazenar código, nome, quantidade, valor de compra e valor de venda de 30 produtos. A listagem pode ser de todos os produtos ou somente de um ao se digitar o código.

```
prog vetor9
 int L,c,t,op,codigo[30],cod, qtde[30];
 string nomes[30];
 real vc[30], vv[30];
 # trecho de entrada de 30 produtos
  para( L<- 0; L <= 29 ; L++)
  {
    imprima "\ndigite ", L + 1, " nome do produto: ";
    leia nomes[ L ] ;
    enquanto( strtam(nomes[L]) >20)
    {imprima "\nNomes com ate 20 caracteres";
     imprima "\n\nDigite ", L + 1, " nome: ";
     leia nomes[ L ] ;
    }
    t<-20-strtam(nomes[L]);
    para(c<-1; c <= t; c++)
    {nomes[L]<-strconcat(nomes[L], " ");}
    imprima "\ndigite codigo do produto: ";
    leia codigo[L] ;
    imprima "\ndigite quantidade: ";
    leia qtde[L] ;
```

```
    imprima "\ndigite valor de compra: ";
    leia vc[L] ;
    imprima "\ndigite valor de venda: ";
    leia vv[L] ;
    }
 # fim do trecho de entrada
 # trecho de saida
   faca
   {
    imprima "\n\n\nMenu\n";
    imprima "\n1-Todos os produtos";
    imprima "\n2-So um produto";
    imprima "\n3-Sair";
    imprima "\nOPCAO: ";
    leia op;
    se(op==1)
    { imprima "\nCodigo\tNome\t\t\tQtde\tValor-compra\tValor-venda";
     para( L<- 0; L <= 29; L++)
     {
      imprima "\n", codigo[L], "\t", nomes[L],"\t",qtde[L], "\t",vc[L],
      "\t\t", vv[L]; # continuacao
     }
     }
     senao
     {se(op==2)
      { c<-0;
       imprima "\nDigite codigo do produto: ";
       leia cod;
       enquanto(cod < >codigo[c] && c<29)
       {c++;}
       se(cod==codigo[c])
       {imprima "\n\n", codigo[c], "\t",nomes[c],"\t",qtde[c], "\t",
       vc[c], "\t", vv[c];} # continuacao
       senao
       { imprima "\n\nCodigo Inexistente";}
       }
       senao
       {se(op==3)
        {saia;}
        senao
        {imprima "\nOpcao Invalida";}}}
     }
     enquanto(op< >3)
  # fim do trecho de saida
 imprima "\n";
fimprog
```

algoritmo 362

Criar um algoritmo que leia dois conjuntos de números inteiros, tendo cada um 10 e 20 elementos e apresente os elementos comuns aos conjuntos. Lem-

bre-se de que os elementos podem se repetir mas não podem aparecer repetidos na saída.

```
prog vetor10
 int vet1[10], vet2[20],vetc[10], i, c, d, L;
 L <- 0;
 imprima "\nentrada de dados do vetor 1 ( 10 elementos)";
 para(i <-0 ; i <= 9; i++)
 { imprima "\nentre com ", i + 1, " elemento: ";
   leia vet1[i];
 }
 imprima "\nentrada de dados do vetor 2 ( 20 elementos)";
 para(i <-0 ; i <= 19; i++)
 { imprima "\nentre com ", i + 1, " elemento: ";
  leia vet2[i];
 }
 para(i <-0 ; i <= 9; i++)
 { vetc[i] <- -999999999;} # inicializando com valores fora do contexto
 para(i <-0 ; i <= 9; i++)
 { c <- 0;
  enquanto(vet1[i] < > vet2[c] && c < 19)
  { c++; }
  se( vet1[i] == vet2[c] )
  { d <- 0;
    enquanto(vet1[i] < > vetc[d]  && d < L)
    {d++; }
    se( d == L)
    { vetc[d]  <- vet1[i]; L++;}
 .}
 }
 se( L < > 0)
 {
   imprima "\n\nElementos comuns\n\n";
   para(i <-0 ; i <= L -1 ; i++)
   { imprima "\n", vetc[i]; }
 }
 senao
 { imprima "\n\nNao existem elementos comuns"; }
 imprima "\n";
fimprog
```

algoritmo 363

Criar um algoritmo que leia vários números inteiros e positivos. A leitura se encerra quando encontrar um número negativo ou quando o vetor ficar completo. Sabe-se que o vetor possui, no máximo, 10 elementos. Gerar e imprimir um vetor onde cada elemento é o inverso do correspondente do vetor original.

```
prog vetor11
 int cont,L;
 real num, v[10],v1[10];
```

```
 cont <-0;
 imprima "\ndigite numero positivo ou 0. para terminar: ";
 leia num;
 enquanto( cont < 10 && num > 0. )
 {
   v[cont] <- num;
   v1[cont] <-  1 / num ;
   cont++;
   imprima "\ndigite numero positivo ou 0. para terminar: ";
   leia num;
 }
 se( cont == 0 )
 {   imprima "\nVetor nao tem dados"; }
 senao
 {  cont--;
    para(L <- 0; L <= cont; L++)
    { imprima "\n", v1[L]; }
 }
 imprima "\n";
fimprog
```

algoritmo 364

Ler um vetor vet de 10 elementos e obter um vetor w cujos componentes são os fatoriais dos respectivos componentes de v.

```
prog vetor12
 int vet[10], w[10], L, f, fat;
 para(L <- 0; L <10 ; L++)
 { imprima "\ndigite elemento ", L + 1,": ";
   leia vet[L];
   w[L] <-1;
   para(f<-2 ; f <= vet[L]; f++)
   { w[L] <- w[L]  * f;}
 }
 imprima "\nVetor fatorial\n";
 para(L <- 0; L <10 ; L++)
 { imprima "\n", w[L]; }
 imprima "\n";
fimprog
```

algoritmo 365

Uma pessoa muito organizada gostaria de fazer um algoritmo para armazenar os seguintes dados de um talonário após a utilização total do mesmo: nº do cheque, valor, data e destino. Sabendo-se que o número de cheques pode ser variável e não ultrapassa 20, construa esse algoritmo de tal maneira que possa gerar um relatório no vídeo.

```
prog vetor13
 string num[20], valor[20], destino[20], data[20], resp;
 int nc, k;
 imprima "\nnumero de cheques do talonario: ";
 leia nc;
 para( k <- 0; k <  nc; k++)
 {  imprima "\nnumero do cheque: ";
    leia num[k];
    imprima "\nvalor do cheque: ";
    leia valor[k];
    imprima "\ndata do cheque ddmmaa: ";
    leia data[k];
    imprima "\ndestino do cheque: ";
    leia destino[k];
 }
 imprima "\nRELACAO dos CHEQUES\n";
 para( k <-0; k < nc; k++)
 { imprima "\nNumero do cheque: ", num[k];
   imprima "\nValor do cheque: ", valor[k];
   imprima "\nData do cheque: ", data[k];
   imprima "\nDestino do cheque: ", num[k];
   imprima "\n\nPressione enter para ver outro cheque: ";
   leia resp; # necessario pois a tela so tem 25 linhas
 }
 imprima "\n";
fimprog
```

algoritmo 366

Criar um algoritmo que armazene em dois vetores nome e profissão de 20 pessoas. Deverá sair uma listagem, no vídeo, que tenha a seguinte forma:

1ª coluna	*41ª coluna*
NOME	*PROFISSAO*
1- ...	*...*
.	
20- ...	*...*

Total de dentistas: - -

```
prog vetor14
 string nomes[20], prof[20];
 int L, tam, c, dent;
 dent <- 0;
  para( L<- 0; L < 20 ; L++)
  { imprima "\nDigite ", L+1,  " nome (com ate 30 caracteres ): " ;
    leia nomes[L] ;
    enquanto(strtam(nomes[L] ) > 30)
    { imprima "\nNome com ate 30 caracteres. Digite ", L+1,  "  nome: ";
```

```
 "ƀoƀ", k +1, "ƀelemento do vetor 2: ", v2[k], "ƀ=ƀ", v4[k];
 }
 imprima "\n";
fimprog
```

algoritmo 369

Criar um algoritmo para gerenciar um sistema de reservas de mesas em uma casa de espetáculo.

A casa possui 30 mesas de 5 lugares cada. O algoritmo deverá permitir que o usuário escolha código de uma mesa (100 a 129) e forneça a quantidade de lugares desejados. O algoritmo deverá informar se foi possível realizar a reserva e atualizar a reserva. Se não for possível, o algoritmo deverá emitir uma mensagem. O algoritmo deve terminar quando o usuário digitar o código 0 (zero) para uma mesa ou quando todos os 150 lugares estiverem ocupados.

```
prog vetor17
 int mesa[30],i,qtde[30],lugares,codigo, lugmesa;
 para(i  <- 0; i < 30 ;i++)
 { mesa[i] <- 100 + i;
  qtde[i] <- 5;
 }
 lugares <- 150;
 imprima "\nentre com codigo (100 - 129) ou 0 para terminar: ";
 leia codigo;
 enquanto(codigo > 0 && lugares < > 0)
 {  i <- 0;
   enquanto(codigo < > mesa[i]  && i < 29)
   { i++; }
   se( codigo == mesa[i] )
   { imprima "\nquantidade de lugares a reservar: ";
     leia lugmesa;
     se( qtde[i] >= lugmesa)
     { qtde[i] <- qtde[i] - lugmesa; lugares <- lugares - lugmesa; }
      senao
      {  imprima "\nnao ha lugares a reservar"; }
    }
    senao
    {  imprima "\ncodigo de mesa invalido"; }
   imprima "\nentre com codigo (100 - 129) ou 0 para terminar: ";
   leia codigo;
 }
 se(lugares == 0)
 {imprima "\nLotacao esgotada";}
 senao
 { imprima "\nLugares vagos\n";
  para(i  <- 0; i < 30 ;i++)
  { se(qtde[i]< >0)
    {imprima "\nmesa: ", mesa[i], " total de lugares: ", qtde[i];
    }
```

```
    }
  }
  imprima "\n";
fimprog
```

algoritmo 370

Criar um algoritmo que realize as reservas de passagem aéreas de uma companhia. Além da leitura do número de vôos e da quantidade de lugares disponíveis, leia vários pedidos de reserva, constituídos do número da carteira de identidade do cliente e do número do vôo desejado.

Para cada cliente, verificar se há possibilidade no vôo desejado. Em caso afirmativo, imprimir o número da identidade do cliente e o número do vôo, atualizando o número de lugares disponíveis. Caso contrário, avisar ao cliente a inexistência de lugares.

```
prog vetor18
 int nv,  i, voos[1000];# numero superestimado
 string nome[1000], id, nvd;
 imprima "\nEntre com o numero de voos:";
 leia nv;
 para (i <- 0; i < nv; i++)
 { imprima "\nEntre com a identificacao  do vo ", i + 1, ": ";
   leia  nome[i];
   imprima "\nEntre com a quantidade de lugares disponiveis no voo ",
   nome[i], ": ";
   leia  voos[i];
 }
 imprima "\nEntre com o numero do voo desejado ou @ para terminar: ";
 leia nvd;
 enquanto( nvd < > "@")
 {  i <- 0;
   enquanto(nvd < > nome[i] && i < nv-1)
   {i++; }
   se(nome[i] == nvd)
   { se(voos[i] > 0)
     { voos[i]--;
       imprima "\nQual o numero da identidade do cliente? ";
       leia id;
       imprima "\nIdentidade:", id, " Voo: ", nvd, "\n";
     }
     senao
     { imprima "\nNao existem mais lugares neste voo.\n"; }
    }
   senao
   { imprima "\nNao existe este voo\n"; }
   imprima "\n\nEntre com o numero do voo desejado ou @ para terminar: ";
   leia nvd;
 }
 imprima "\n";
fimprog
```

algoritmo 371

Criar um algoritmo que leia dois conjuntos de números inteiros, tendo cada um 10 e 20 elementos e apresentar os elementos que não são comuns aos dois conjuntos.

```
prog vetor19
 int vet1[10], vet2[20],vetc[30], i, c, d, L;
 L <- 0;
 imprima "\nentrada de dados do vetor 1 ( 10 elementos)";
 para(i <-0 ; i <= 9; i++)
 { imprima "\nentre com ", i + 1, " elemento: ";
  leia vet1[i];
 }
 imprima "\nentrada de dados do vetor 2 ( 20 elementos)";
 para(i <-0 ; i <= 19; i++)
 { imprima "\nentre com ", i + 1, " elemento: ";
  leia vet2[i];
 }
 para(i <-0 ; i <= 29; i++)
 { vetc[i] <- -999999999;} # inicializado com valores fora do contexto
 para(i <-0 ; i <= 9; i++)
 { c <- 0;
  enquanto(vet1[i] < > vet2[c] && c < 19)
  { c++; }
  se( vet1[i] < > vet2[c])
  { d <- 0;
    enquanto(vet1[i] < > vetc[d]  && d < L)
    {d++; }
    se( d == L)
    { vetc[d]  <- vet1[i]; L++;}
   }
 }
 para(i <-0 ; i <= 19; i++)
 { c <- 0;
  enquanto(vet2[i] < > vet1[c] && c < 9)
  { c++; }
  se( vet2[i] < > vet1[c])
  { d <- 0;
    enquanto(vet2[i] < > vetc[d]  && d < L)
    {d++; }
    se( d == L)
    { vetc[d]  <- vet2[i]; L++;}
   }
 }
 se( L < > 0)
 {
  imprima "\n\nElementos nao comuns";
```

```
  para(i <-0 ; i <= L -1 ; i++)
  { imprima "\n", vetc[i]; }
 }
 senao
 { imprima "\n\nNao existem elementos nao comuns" ;}
 imprima "\n";
fimprog
```

algoritmo 372

Num torneio de futsal, rodada simples, inscreveram-se 12 times. Armazenar os nomes dos times e imprimir a tabela de jogos.

```
prog vetor20
 int L, c,t;
  string  times[12];
 #trecho de entrada
  para( L<- 0;L <= 11; L++)
  {
     imprima "\n\nDigite ", L + 1, " time: ";
     leia times[L];
  enquanto( strtam(times[L]) >20)
    {imprima "\nNomes com ate 20 caracteres";
    imprima "\n\nDigite ", L + 1, " time: ";
    leia times[ L ] ;
    }
    t<-20-strtam(times[L]);
    para(c<-1; c <= t; c++)
    {times[L]<-strconcat(times[L], "ȸ");}
   }
  imprima "\n Rodada Simples";
  para( L<- 0;L <= 10; L++)
  {
   para( c<- L +1;c <= 11; c++)
    {
      imprima "\n", times[L ],"\t",times[c ];
   }
 }
 imprima "\n";
fimprog
```

algoritmo 373

Entrar com nomes de cinco times de futebol e armazená-los em um vetor de nome TIMES. Imprimir uma tabela para rodada dupla.

```
prog vetor21
 int L, c,t;
 string  times[5];
 para( L<- 0;L <= 4; L++)
```

```
{
    imprima "\ndigite nome do time:";
    leia times[L];
    enquanto( strtam( times[L]) >20)
    { imprima "\nNomes com ate 20 caracteres";
      imprima "\n\nDigite ", L + 1, " time: ";
      leia times[ L ] ;
    }
   t<-20-strtam(times[L]);
   para(c<-1; c <= t; c++)
   {times[L]<-strconcat(times[L], "b");}
  }
 imprima "\n Rodada Dupla";
 para( L<- 0;L <= 4; L++)
 {
  para( c<- 0;c <= 4; c++)
   {
      se(L < > c)
      {   imprima "\n", times[L ],"\t",times[ c ];  }
  }
 }
imprima "\n";
fimprog
```

algoritmo 374

Entrar com números inteiros em um vetor A [50]. Gerar e imprimir o vetor B onde cada elemento é o quadrado do elemento, na respectiva posição, do vetor A.

```
prog vetor22
 int L,a[50], b[50];
 #trecho de entrada para 50 numeros
 para( L<- 0;L <= 49 ; L++)
 {
  imprima "\ndigite numero :";
  leia a[L ];
  b[L] <- a[L ] ^ 2;
 }
 #trecho de saida
 imprima "\nOriginal\tQuadrado";
 para( L<- 0;L <= 49; L++)
 {   imprima "\n",a[L], "\t\t",b [L ] ;}
 imprima "\n";
fimprog
```

algoritmo 375

Entrar com números reais para dois vetores A e B de dez elementos cada. Gerar e imprimir o vetor diferença.

```
prog vetor23
 int L;
 real a[30], b[30], dif[30];
 #trecho de entrada para 30 numeros
 para( L<- 0;L <= 29; L++)
  {
    imprima "\ndigite elemento ",L+1, " do vetor A:";
    leia a[L ];
  }
   imprima "\n\n";
 para( L<- 0;L <= 29 ; L++)
  {
    imprima "\ndigite elemento ",L+1, " do vetor B:";
    leia b[L ];
    #Gera o vetor diferenca
    dif [L] <- a[L ] - b[L] ;
  }
 #trecho de saida
  imprima "\nVetor A\t\tVetor B\t\tVetor Diferenca";
  para( L<- 0;L <= 29; L++)
 {
   imprima "\n",a[L], "\t\t",b [L], "\t\t\t", dif[L]   ;
  }
 imprima "\n";
fimprog
```

algoritmo 376

Criar um algoritmo que leia um vetor A de dez valores e construa outro vetor B, da seguinte forma:

Ex.: Vetor A	*3*	*8*	*4*	*2*	...	*5*
Vetor B	*9*	*4*	*12*	*1*	...	*15*

```
prog vetor24
 int L ;
 real a[10], b[10];
 #vetor com 10 elementos
 para( L<- 0;L <= 9 ; L++)
 {
    imprima "\ndigite numero :";
    leia a[L ];
    se(L   % 2 == 0)
    { b[L ] <- a[L ] / 2; }
     senao
     { b[L ] <- a[L ] * 3; }
 }
```

```
 imprima "\n\n Vetores\n";
 para( L<- 0;L <= 9; L++)
 {    imprima a [L ], "  " ;  }
 imprima "\n";
 para( L<- 0;L <= 9; L++)
 {    imprima b [L ], "  " ;  }
 imprima "\n";
fimprog
```

algoritmo 377

Criar um algoritmo que leia dois vetores A e B, contendo, cada um, 25 elementos inteiros. Intercale esses dois conjuntos (A[1] / B[1] / A[2] / B[2] /..), formando um vetor V de 50 elementos. Ordene de forma decrescente. Imprima o vetor V.

```
prog vetor25
 int L,  a[25], b[25], inter[50];
 #25 elementos cada
 para( L<- 0;L <= 24; L++)
 {
    imprima "\ndigite ", L+1," elemento do vetor A :";
    leia a[L ];
 }
 para( L<- 0;L <= 24; L++)
 {
    imprima "\ndigite ", L+1," elemento do vetor B:";
    leia b[L ];
    #Gera o vetor intercalado
    inter[2 * L] <- a[L ] ;
    inter[2 * L + 1] <- b[L ] ;
 }
 imprima "\nVetor Intercalado";
 para( L<- 0;L <= 49; L++)
 {
    imprima "\n", inter[L];
 }
 imprima "\n";
fimprog
```

algoritmo 378

Entrar com vários números, até digitar o número 0. Imprimir quantos números iguais ao último número foram lidos. O limite de números é 100.

```
prog vetor26
 int L ,c,d,  num[100];
 # trecho de entrada de 100
 L<- 0;
 imprima "\ndigite ", L + 1, " numero ou 0 para terminar: ";
```

```
leia num[ L ] ;
enquanto(L<=98 && num[L]< >0)
{ L++;
 imprima "\ndigite ", L + 1, " numero ou 0 para terminar: ";
 leia num[ L ] ;
}
# fim do trecho de entrada
se(num[L] ==0)
{L--;}
c<-0;
para(d<-0; d<=L-1;d++)
{se(num[d] ==num[L])
 {c++;}
}
# trecho de saida
imprima "\nTotal de numeros iguais ao ultimo: ", c;
# fim do trecho de saida
imprima "\n";
fimprog
```

algoritmo 379

Fazer um algoritmo para ler um conjunto de 100 números reais e informar:

1. *quantos números lidos são iguais a 30*
2. *quantos são maior que a média*
3. *quantos são iguais à média*

```
prog vetor27
 real num[100],soma, media;
 int cm,cmm, c30, L;
 soma <-0.;
 c30 <-0;
 para(L <-0;L<=99;L++)
 {
  imprima "\ndigite numero: ";
  leia num[L];
  soma <-soma +num[L];
  se(num[L]==30.)
  {c30++;}
 }
 media <-soma/100;
 cm <-0;
 cmm <-0;
 para(L <-0;L<=99;L++)
 {
  se(num[L] > media)
  {cmm++;}
  senao
```

```
  {se(num[L]==media)
  {cm++;}}
 }
 imprima "\nTotal iguais a 30: ",c30;
 imprima "\nTotal iguais a media: ",cm;
 imprima "\nTotal maiores que a media: ",cmm;
 imprima "\n";
fimprog
```

algoritmo 380

Criar um algoritmo que leia um conjunto de 30 valores inteiros, armazene-os em um vetor e escreva-os ao contrário da ordem de leitura.

```
prog vetor28
 int L, num[30];
 para(L<-0; L<=29;L++)
 {imprima "\ndigite numero: ";
 leia num[L];
 }
 imprima "\n\n\n";
 imprima "\n Vetor ao Contrario\n";
 para(L<-29; L>=0;L--)
 {imprima "\n",num[L];}
 imprima "\n";
fimprog
```

algoritmo 381

Armazenar dez nomes em um vetor NOME e imprimir uma listagem numerada e ordenada.

```
prog vetor29
 int L, c;
 string nomes[10], aux;
 para( L<- 0; L <= 9 ; L++)
 {
  imprima "\n nome: " ;
  leia nomes[L] ;
 }
 para( L<- 0; L <= 8 ; L++)
 {
  para( c<- L + 1; c <= 9 ; c++)
  {
    se (  nomes[L] > nomes[c] )
    {
      aux <- nomes[L];
      nomes[L] <- nomes[c ];
      nomes[c] <- aux;
```

```
      }
    }
  }
  imprima "\n\n\n";
  imprima "\n Relação  dos nomes ordenados\n " ;
  para( L<- 0; L <= 9 ; L++)
  {
      imprima "\n", L +1, " - ", nomes[L] ;
  }
  imprima "\n";
fimprog
```

algoritmo 382

Entrar com dados para um vetor VET do tipo inteiro com 20 posições, onde podem existir vários elementos repetidos. Gere o vetor VET1 que também será ordenado e terá somente os elementos do vetor VET que não são repetidos.

```
prog vetor30
 int L, c,vet[20], vet1[20], aux;
 para( L<- 0; L <= 19 ; L++)
 {
  imprima "\n numero: " ;
  leia vet[L] ;
 }
 para( L<- 0; L <= 18 ; L++)
 {
  para( c<- L + 1; c <= 19 ; c++)
  {  se (  vet[L] > vet[c] )
     {  aux <- vet[L];   vet[L] <- vet[c ];   vet[c] <- aux;    }
   }
 }
 imprima "\n\n\n";
 imprima "\n Relação  dos numeros ordenados\n " ;
 para( L<- 0; L <= 19 ; L++)
 {    imprima "\n", L +1, " - ", vet[L] ;}
 vet1[0]<- vet[0];
 c<-1;
 para( L<- 1; L <= 19 ; L++)
 { se(vet[L] < > vet1[c-1])
  {   vet1[c] <- vet[L];   c++;  }
 }
 imprima "\n\n\n";
 imprima "\n Relação  dos numeros nao repetidos\n " ;
 para( L<- 0; L <= c-1 ; L++)
 {   imprima "\n", L +1, " - ", vet1[L] ;}
 imprima "\n";
fimprog
```

algoritmo 383

Criar um algoritmo que leia os elementos de um vetor com 20 posições e escreva-o. Em seguida, troque o primeiro elemento pelo último, o segundo pelo penúltimo, o terceiro pelo antepenúltimo, e assim sucessivamente. Mostre o vetor depois das trocas.

```
prog vetor31
 int L, c,vet[20], aux;
 para( L<- 0; L <= 19 ; L++)
 {  imprima "\n numero: " ;  leia vet[L] ;}
 para( L<- 0; L <= 9 ; L++)
 {  aux <- vet[L];  vet[L] <- vet[19-L];   vet[19-L] <- aux; }
 imprima "\n\n\n";
 imprima "\nVetor trocado\n " ;
 para( L<- 0; L <= 19 ; L++)
 { imprima "\n", L +1, " - ",vet[L];}
 imprima "\n";
fimprog
```

algoritmo 384

Em um concurso público inscreveram-se 5.000 candidatos para 100 vagas. Cada candidato fez 3 provas, tendo cada uma pesos 2, 3 e 5 respectivamente, na ordem em que foram feitas. Fazer um algoritmo que leia nome, matrícula e os pontos obtidos pelos candidatos em cada prova; apresentar a classificação, a matrícula e o nome dos candidatos aprovados, ordenados pela classificação.

```
prog vetor32
 int L, c,tam,p1[5000], p2[5000],p3[5000], pontos[5000], mat[5000],auxn;
 string nomes[5000], auxc;
 para( L<- 0; L <= 4999 ; L++)
 {
   imprima "\n nome: " ;
   leia nomes[L] ;
   enquanto(strtam(nomes[L])>30)
   { imprima "\nNomes com ate 30 caracteres. Digite nome: ";
     leia nomes[L];}
   tam <-strtam(nomes[L]);
   se(tam<30)
   { para(c<-0;c<=30 - tam; c++)
     { nomes[L]<-strconcat(nomes[L], "ƀ");}}
   imprima "\n matricula: " ;  leia mat[L] ;
   imprima "\n pontos da 1 prova: " ;  leia p1[L] ;
   imprima "\n pontos da 2 prova: " ;  leia p2[L] ;
   imprima "\n pontos da 3 prova: " ;  leia p3[L] ;
   pontos[L]<-p1[L]+p2[L]+p3[L];
 }
```

```
para( L<- 0; L <= 4998 ; L++)
{
 para( c<- L + 1; c <= 4999 ; c++)
 {
   se (  pontos[L] < pontos[c] )
   {
     auxn <- pontos[L];
     pontos[L] <- pontos[c ];
     pontos[c] <- auxn;
     auxn <- p1[L];
     p1[L] <- p1[c ];
     p1[c] <- auxn;
     auxn <- p2[L];
     p2[L] <- p2[c ];
     p2[c] <- auxn;
     auxn <- p3[L];
     p3[L] <- p3[c ];
     p3[c] <- auxn;
     auxn <- mat[L];
     mat[L] <- mat[c ];
     mat[c] <- auxn;
     auxc <- nomes[L];
     nomes[L] <- nomes[c ];
     nomes[c] <- auxc;
   }
  }
}
imprima "\n\n\n";
imprima "\nCLAS.\tMAT\tNOME\t\t\t\tP1\tP2\tP3\tSOMA\n " ;
para( L<- 0; L <= 99 ; L++)
{
 imprima "\n",L +1,"\t",mat[L], "\t",nomes[L],"\t", p1[L], "\t", p2[L],
 "\t", p3[L], "\t",pontos[L];
}
imprima "\n";
fimprog
```

algoritmo 385

No vestibular de uma universidade, no curso de Informática, inscreveram-se 1.200 pessoas. Criar um algoritmo que leia o gabarito da prova que tinha 100 questões, sendo o valor de cada questão igual a 1 ponto. Exiba o número de inscrição, o nome e as 100 respostas de cada candidato. O algoritmo deverá imprimir: o número de inscrição, o nome e a nota de cada candidato.

Só é necessário guardar a soma dos pontos de cada candidato, o número de inscrição e o nome.

```
prog vetor33
 int L,c, soma,tam, insc[1200],pontos[100];
 string nomes[1200], gab[100], resp;
 para(L<-0;L<=99;L++)
 {imprima "\nDigite gabarito (a/b/c/d/e) da questao ", L+1,": ";
 leia gab[L];}
 para(L<-0;L<=1199;L++)
 {imprima "\nDigite numero de inscricao do candidato ", L+1, ": ";
  leia insc[L];
  imprima "\nDigite nome do candidato: "; leia nomes[L];
  enquanto(strtam(nomes[L]) > 30)
  {imprima "\nNomes com ate 30 caracteres. Digite nome: ";
   leia nomes[L];}
   tam <-strtam(nomes[L] );
   se(tam<30)
   {para(c<-0;c<=30 - tam; c++)
     {nomes[L]<-strconcat(nomes[L], "ƀ"); }
 }
 soma <-0;
 para(c<-0;c<=99;c++)
 {imprima "\nDigite resposta da ", c+1," questao: ";
  leia resp;
  se(resp==gab[c])
  { soma++;}
  }
  pontos[L]<-soma;
 }
 imprima "\n\n\n";
 imprima "\nINSC.\tNOME\t\t\t\tPONTOS\n";
 para(L<-0;L<=1199;L++)
 {imprima "\n",insc[L], "\t",nomes[L],"\t",pontos[L];}
 imprima "\n";
fimprog
```

algoritmo 386

Suponha três vetores de 30 elementos cada, contendo: nome, endereço, telefone. Fazer um trecho que se possa buscar pelo nome e imprimir todos os dados.

```
prog vetor34
 int L ;
 string nome, nomes [30], end[30], tel[30];
 para( L<- 0; L <= 29 ; L++)
 {
   imprima "\n nome ", L+1, ": " ;   leia nomes[L] ;
   imprima "\n endereco: " ;   leia end[L] ;
   imprima "\n telefone: " ;   leia tel[L] ;
 }
```

```
# entra com nome para procura
imprima "\n Digite nome para procura: " ;
leia nome;
L <-0;
enquanto ( L<29 &&  nome< >nomes[L]  )
{  L++;  }
se (nome ==nomes[L] )
{
 imprima "\nnome     : ", nomes[L];
 imprima "\nendereco: ", end[L];
 imprima "\ntelefone: ", tel[L];
}
senao
{  imprima "\nnao encontrei" ; }
imprima "\n";
fimprog
```

algoritmo 387

Fazer um algoritmo que leia a matrícula e a média de 1.000 alunos. Ordene da maior nota para a menor e imprima uma relação contendo todas as matrículas e médias.

```
prog vetor35
 int L, c, mat[1000],auxi ;
 real media[1000], auxr;
 para( L<- 0; L <= 999 ; L++)
 {
  imprima "\n matricula do aluno ",L+1,": " ;
  leia mat[L] ;
  imprima "\n media: " ;
  leia media[L] ;
 }
 para( L<- 0; L <= 998 ; L++)
 {
  para( c<- L + 1; c <= 999; c++)
  {
    se (  media[L] < media[c] )
    {
      auxr <- media[L];
      media[L] <- media[c];
      media[c] <- auxr;
      auxi <- mat[L];
      mat[L] <- mat[c];
      mat[c] <- auxi;
    }
  }
 }
```

```
 imprima "\nORDENADOS\n";
 imprima "\nMATRICULA\tMEDIA\n " ;
 para( L<- 0; L <= 999 ; L++)
 {   imprima "\n", mat[L], "\t\t",media[L] ;}
 imprima "\n";
fimprog
```

algoritmo 388

Criar um algoritmo que armazene 10 números em um vetor. Na entrada de dados, o número já deverá ser armazenado na sua posição definitiva em ordem decrescente. Imprimir o vetor logo após a entrada de dados.

Exemplo:

entrada	*vetor*
7	*7*

entrada	*vetor*
7	*9*
9	*7*

entrada	*vetor*
7	*9*
9	*7*
2	*2*

entrada	*vetor*
7	*12*
9	*9*
2	*7*
12	*2*
...	*...*

```
prog vetor36
 int L, c,d,n, a[10];
 para( L<- 0; L <= 9 ; L++)
 {a[L]<-0; }
 para( L<- 0; L <= 9 ; L++)
 {
  imprima "elemento do vetor A[", L + 1, "]: ";  leia  n;
  c<-0;
  enquanto(n<=a[c])
  {c++;}
  se(L>0)
  {para(d<-L;d>=c+1;d--)
   {a[d]<-a[d-1];}
   }
   a[c]<-n;
 }
 imprima "\n";
 imprima "\n\n\nVetor Ordenado\n";
 para( L<- 0; L <= 9 ; L++)
 {
  imprima "\na[",L + 1, "]- ", a[L];
 }
 imprima "\n";
fimprog
```

algoritmo 389

Criar um algoritmo que receba a temperatura média de cada mês do ano, em centígrados, e armazene essas temperaturas em um vetor; imprimir as temperaturas de todos os meses, a maior e a menor temperatura do ano e em que mês aconteceram.

```
prog vetor37
 int L, maiorM, menorM;
 real temp[12];
 maiorM <- 0;
 menorM <- 0;
 para( L<- 0; L <= 11 ; L++)
 {
   imprima "\n temperatura do mes ", L+1,": " ;
   leia temp[L] ;
   se (  temp[L] > temp[maiorM] )
    {   maiorM <- L;    }
   senao
   {
      se (  temp[L] < temp[menorM] )
        {     menorM <- L;  }
    }
 }
 imprima "\n\n\n";
 imprima "\n Relação  das temperaturas\n " ;
 para( L<- 0; L < 12 ; L++)
 {    imprima "\nmes: ", L +1, " - temperatura ", temp[L] ;}
 imprima "\nMAIOR TEMPERATURA - ", temp[maiorM]," no mes: ", maiorM + 1;
 imprima "\nMENOR TEMPERATURA - ", temp[menorM]," no mes: ", menorM + 1;
 imprima "\n";
fimprog
```

algoritmo 390

Ler nome, CPF e profissão de 100 pessoas. Imprimir qual(ais) a(s) profissão(ões) que mais se repete(m) e quantas pessoas têm essa(s) profissão(ões).

```
prog vetor38
 int L,c,x, conta[100];
 string nome[100],cpf[100],prof[100],cprof[100];
 para( L<-0; L <= 99;L++)
 {
   imprima "\ndigite nome: ";    leia nome[L];
   imprima "\ndigite CPF: ";   leia cpf[L];
   imprima "\ndigite profissao somente no sexo masculino: ";
   leia prof[L];
```

```
}
/*impressao das profissoes de todas as pessoas so para entendimento*/
imprima "\n";
 para( L<-0;L <= 99;L++)
{    imprima "\n",L+1," - ",prof[L]; }
/*inicializa os vetores */
para( L<-0;L <= 99; L++)
{   conta[L] <-0; cprof[L]<-""; }
cprof[0] <- prof[0];
conta[0]++;
c <-1;
para( L <- 1;L <= 99;L++)
{
x <-0;
enquanto(prof[L] < > cprof[x]  &&  x < c)
{ x++; }
se( x == c)
{  cprof[c] <- prof[L];  conta[c]++;  c++;}
senao
{conta[x]++;}
}
imprima "\n";
/*impressao das profissoes so para entendimento*/
imprima "\nPROFISSOES(AO) ENCONTRADA(S)";
para( L <-0;L <= c-1; L++)
{
  imprima "\n",L+1, " - ",cprof[L];
}
/*descobre o maior*/
imprima "\n";
x <- 0;
para(L <- 1;L <= c-1; L++)
{
   se( conta[L] > conta[x])
   { x <- L; }
}
/*imprime os maiores*/
imprima "\nPROFISSAO(OES)  E QUANTIDADE(S)";
para( L <- 0;L <= c-1; L++)
{
   se(conta[L] == conta[x])
   {imprima "\n", cprof[L]," - ",conta[L];}
}
imprima "\n";
fimprog
```

algoritmo 391

Criar um algoritmo que leia um vetor de 30 números inteiros e imprima o número de elementos da maior sublista ordenada crescentemente.

Exemplo 1: \ 8 \ 9 \ 1 \ 7 \ 8 \ 17 \ 3 \ ....
Maior sublista: 1 \ 7 \ 8 \ 17 \ logo o tamanho é 4.
Exemplo 2: \ 18 \ 9 \ 5 \ 3 \ 1 ....
Maior sublista: não existe mais de um elemento, logo o tamanho é 1.

```
prog vetor39
 int M[30], L, C, Y;
 para( L <- 0;  L <= 29 ; L++ )
 {   imprima "\nDigite o elemento ", L + 1,":  ";   leia M[L]; }
 Y <- 1;
 C <- 0;
 para( L <- 0;  L <= 28 ; L++ )
 {
    se( M[L + 1] > M[L] )
   {    Y ++;  }
   senao
   {
        se( Y > C )
        {    C <-  Y;  }
        Y <-  1;
   }
 }
 se( C > Y )
 {     imprima "\ntotal: ",  C;}
 senao
 {     imprima "\ntotal: ",  Y;}
 imprima "\n";
fimprog
```

algoritmo 392

Criar um algoritmo que implemente o seguinte menu de opções:

- *Ler notas e nomes de 100 candidatos;*
- *Exibir média geral de todos os candidatos;*
- *Exibir uma lista com o nome e a nota de todos os candidatos em ordem decrescente de nota;*
- *Ler um nome e buscar esse candidato imprimindo o nome e sua nota; caso não seja encontrado candidato com o nome lido, imprimir a mensagem "Candidato não encontrado".*

```
prog vetor40
 string nomes[100], auxnome, nomep;
```

```
real notas[100], mediag, auxnota;
int opcao, c, L;
faca
{
  imprima "\n\n\n\t MENU";
  imprima "\n 1 - Ler Nomes e Notas";
  imprima "\n 2 - Media Geral";
  imprima "\n 3 - Classificacao";
  imprima "\n 4 - Busca Candidato";
  imprima "\n 5 - Sair";
  imprima "\n\t OPCAO: ";
  leia opcao;
  se(opcao == 1)
  { para(c <-0 ;c <= 99; c++)
    { imprima "\n Digite o nome: "; leia nomes[c];
      imprima "\n Digite a nota com ponto: "; leia notas[c];
    }
  }
  senao
   { se(opcao == 2)
      { mediag <- 0.;
        para(c <- 0; c <= 99; c++)
         { mediag <- mediag + notas[c];
         }
         imprima "\n Media Geral = ", mediag/100;
      }
    senao
      { se(opcao == 3)
           { para( L<- 0; L <= 98 ; L++)
             { para( c<- L + 1; c <= 99 ; c++)
                { se(  notas[L] < notas[c] )
                   { auxnota <- notas[L];
                      notas[L] <- notas[c];
                      notas[c] <- auxnota;
                      auxnome <- nomes[L];
                      nomes[L] <- nomes[c];
                      nomes[c] <- auxnome;
                   }
                }
            }
          imprima "\n Relacao em ordem decrescente de nota";
          para( c<- 0; c <= 99 ; c++)
           { imprima "\n", c+1, " - Nome: ", nomes[c], "\t Nota: ",
             notas[c]; }
         }
         senao
         { se(opcao == 4)
            { imprima "\n Digite o nome a ser procurado"; leia nomep;
```

```
                L <- 0;
                enquanto( L < 99 &&  nomep < > nomes[L]  )
                { L <-L + 1; }
                se(nomep ==nomes[L] )
                { imprima "\n NOME: ", nomes[L], "\nNOTA;", notas[L];  }
                senao
                {  imprima "\n Nao encontrei" ;  }
                imprima "\n";
            }
            senao
            { se(opcao <> 5 )
              { imprima "\n Opcao Invalida";}
            }
          }
        }
    }
 }
 enquanto(opcao< >5)
 imprima "\n";
fimprog
```

algoritmo 393

Um sistema de controle de estoque armazena nome, quantidade em estoque e preço unitário de 40 mercadorias. Fazer um menu que exiba as seguintes opções:

> *MENU*
>
> *1 – Cadastra mercadorias*
>
> *2 – Exibe valor total em mercadorias da empresa*
>
> *3 – Sai*
>
> *OPCAO:*

```
prog vetor41
 string nomes[40];
 int op, x, y, cadastrou,estoque[40];
 real auxhoras, auxiliar,preco[40], soma;
 cadastrou <- 0;
 faca
 {
  imprima "\n1 - Cadastra mercadorias";
  imprima "\n2 - Exibe valor total em mercadorias da empresa";
  imprima "\n3 - Sai";
  imprima "\nDigite sua opcao:";
  leia op;
  se(op == 1)
  { para( x <- 0; x <= 39; x++)
    { imprima "\ndigite nome da mercadoria: ";
```

```
      leia nomes[x];
      imprima "\ndigite quantidade em estoque: ";
      leia estoque[x];
      imprima "\ndigite preco da mercadoria: ";
      leia preco[x];
   }
   cadastrou <- 1;
 }
 senao
 { se(op == 2)
  { se(cadastrou == 0)
    {  imprima "\nNao ha  dados cadastrados, escolha opcao 1";}
    senao
    { soma <- 0.;
      para( x <- 0; x <= 39; x++)
      {  soma <- soma + estoque[x] * preco[x]; }
        imprima "\nvalor total em estoque = ",soma;
     }
   }
   senao
   { se(op == 3)
     { imprima "\nEncerrar algoritmo";}
     senao
     { imprima "\nOpcao invalida";}
    }
  }
  imprima "\n\n";
 }
 enquanto(op < > 3)
 imprima "\n";
fimprog
```

algoritmo 394

Criar um algoritmo que possa armazenar nome, duas notas e média de 50 alunos. A média será calculada segundo o critério: peso 3 para a primeira nota e peso 7 para a segunda. A impressão deverá conter nome, duas notas e a média.

ESCOLA VIVA
1 - Entrar nomes
2 - Entrar 1ª nota
3 - Entrar 2ª nota
4 - Calcular média
5 - Listar no display
6 - Sair
opcao

```
prog vetor42
 int L, c,tam, flag,flag1, flag2;
 real nota1[50], nota2[50], media[50];
 string nomes[50],op;
 flag <-0;
 flag1 <-0;
 flag2 <-0;
 faca
 {imprima "\n\n\n";
 imprima "\n MENU\n";
 imprima "\n1 - ENTRAR NOMES";
 imprima "\n2 - ENTRAR 1 NOTA";
 imprima "\n3 - ENTRAR 1 NOTA";
 imprima "\n4 - CALCULAR MEDIA";
 imprima "\n5 - LISTAR NO DISPLAY";
 imprima "\n6 - SAIR";
 imprima "\nOPCAO:";
 leia op;
 se(op=="1")
 {flag<-1;
  para( L<- 0; L <= 49 ; L++)
  {
  imprima "\nDigite ", L+1,  " nome (com ate 30 caracteres e todas as
  letra maiusculas): " ; # na mesma linha anterior
  leia nomes[L] ;
  enquanto(strtam(nomes[L]) >30)
  {imprima "\nNome com ate 30 caracteres. Digite ", L+1,  "  nome (todas
  as letras maiusculas): ";#  na mesma linha anterior
   leia nomes[L];}
   tam <-strtam(nomes[L]);
   se(tam <30)
   {para(c<-0;c<=30 - tam; c++)
    {nomes[L]<-strconcat(nomes[L], "ƀ");}}
   }
  }
  senao
  { se(op=="2")
    { se(flag==0)
      {imprima "\n NAO TEM NOME CADASTRADO";}
      senao
      {
       para( L<- 0; L <= 49 ; L++)
       { imprima "\nDigite 1 nota: ";
         leia nota1[L];
       }flag1<-1;
       }
      }
      senao
```

```
      { se(op=="3")
        { se(flag==0)
          {imprima "\n NAO TEM NOME CADASTRADO";}
          senao
          {
           para( L<- 0; L <= 49 ; L++)
           { imprima "\nDigite 2 nota: ";
             leia nota2[L];
           }flag2<-1;
          }
        }
       senao
        {se(op=="4")
         {   se(flag==0 || flag1==0 || flag2==0)
             {imprima "\n NEM TODOS OS DADOS ESTAO CADASTRADOS";}
             senao
             {  para( L<- 0; L <= 49 ; L++)
                { media[L]<-(3*nota1[L] + 7*nota2[L])/10;       }
             }
          }
          senao
           {se(op=="5")
            { se(flag==0 || flag1==0 || flag2==0)
              {imprima "\n NEM TODOS OS DADOS ESTAO CADASTRADOS";}
              senao
              { imprima "\nNOME\t\t\t\tNOTA1\tNOTA2\tMEDIA\n";
               para( L<- 0; L <= 49 ; L++)
               { imprima "\n", nomes[L], "\t", nota1[L],"\t", nota2[L],
                 "\t", media[L];
               }
              }
             }
             senao
            {se(op=="6")
             {saia;}
             senao
             {imprima "\nOPCAO NAO DISPONIVEL";
             } } } } } }
 }
 enquanto(op< >"6")
 imprima "\n";
fimprog
```

algoritmo 395

Criar um algoritmo que possa armazenar nomes e salários de 15 pessoas. A listagem deverá conter nomes e salários, tabulados.

MENU

1 – INSERIR
2 – ORDENAR
3 – LISTAR
4 – PROCURAR
5 – SAIR
OPCAO:

```
prog vetor43
 int L, c,tam, flag;
 real sal[15], auxn;
 string nomes[15], auxc,op, nomep;
 flag <-0;
 faca
 {imprima "\n\n\n";
 imprima "\n MENU\n";
 imprima "\n1 - INSERIR";
 imprima "\n2 - ORDENAR";
 imprima "\n3 - LISTAR";
 imprima "\n4 - PROCURAR";
 imprima "\n5 - SAIR";
 imprima "\nOPCAO:";
 leia op;
 se(op=="1")
 {flag<-1;
  para( L<- 0; L <= 14 ; L++)
  {
      imprima "\nDigite ", L+1,  " nome : " ;
      leia nomes[L] ;
      enquanto(strtam(nomes[L] ) > 30)
      {  imprima "\nNome com ate 30 caracteres. Digite ", L+1,  "  nome: ";
           leia nomes[L]; }
      tam <- strtam(nomes[L]);
       se(tam < 30)
       {  para(c<-0;c<=30 - tam; c++)
           {nomes[L]<-strconcat(nomes[L], "ƀ");}}
           imprima "\n salario (maior do que 0.): " ;
           leia sal[L] ;
           enquanto(sal[L] < 0.)
           {imprima "\nNAO EXISTE SALARIO NEGATIVO. Digite salario (maior
           do que 0.): ";
             leia sal[L]; }
       }
  }
  senao
  { se(op=="2")
    { se(flag==0)
```

```
{imprima "\n NAO TEM DADOS CADASTRADOS";}
senao
{
 para( L<- 0; L <= 13 ; L++)
 {
   para( c<- L + 1; c <= 14 ; c++)
   {
    se (  nomes[L] > nomes[c] )
    {
     auxc <- nomes[L];
     nomes[L] <- nomes[c];
      nomes[c] <- auxc;
      auxn <- sal[L];
      sal[L] <- sal[c];
      sal[c] <- auxn;
    } }  } }  }
senao
{ se(op=="3")
  { se(flag==0)
    {imprima "\n NAO TEM DADOS CADASTRADOS";}
    senao
    {
     imprima "\n\n\n";
     imprima "\n   NOME\t\t\t\t\tSALARIO\n " ;
     para( L<- 0; L <= 14 ; L++)
     {
      imprima "\n",L +1," - ",nomes[L],"\t", formatar(sal[L],2);
     }
    }
   }
  senao
  {se(op=="4")
   { se(flag==0)
    {imprima "\n NAO TEM DADOS CADASTRADOS";}
    senao
    { imprima "\nDigite nome para procura: ";
      leia nomep;
      enquanto(strtam(nomep)>30)
      {imprima "\nNome com ate 30 caracteres. Digite nome: ";
       leia nomep;}
       tam <-strtam(nomep);
       se(tam <30)
       {para(c<-0;c<=30 - tam; c++)
       {nomep<-strconcat(nomep, " ");}}
      c<-0;
      enquanto(nomep< >nomes[c] && c < 14)
      {c++;}
      se(nomep==nomes[c])
      {imprima "\nnome: ",nomes[c];
```

```
            imprima "\nsalario R$ ",formatar(sal[c],2);}
            senao
            {imprima "\nNOME NAO ENCONTRADO";}
            }
           }
           senao
           {se(op=="5")
            {saia;}
            senao
            {imprima "\nOPCAO NAO DISPONIVEL";
            } } } } }
}
 enquanto(op< >"5")
 imprima "\n";
fimprog
```

algoritmo 396

Criar um algoritmo que entre com nome e a matrícula para as disciplinas de Programação I e Sistemas de Informação. Cada disciplina tem 100 vagas. Após a entrada de dados, aparecerá o menu a seguir:

LISTA *1 – Todos de Programação I* *2 – Todos de Sistemas de Informação* *3 – Todos que fazem as duas* *4 – Sair* *Opcao:*

1. *Ao se digitar 1, sairão todos os nomes e respectivas matrículas.*
2. *Ao se digitar 2, sairão todos os nomes e respectivas matrículas.*
3. *Ao se digitar 3, sairão todos os nomes e respectivas matrículas. A PROCURA DEVERÁ SER FEITA PELA MATRÍCULA.*
4. *Ao se digitar 4, aparecerá a mensagem: FECHANDO.*
5. *Qualquer outro número, aparecerá a mensagem: OPÇÃO INVÁLIDA.*

```
prog vetor44
 string nomesi[100], nomeprog[100];
 int vetsi[100]  ,vetprog[100], I, J, OP,  C;
 imprima "\nEntre com os dados dos alunos de Programacao I";
 para( I <- 0; I <= 99; I++)
 {  imprima "\n NOME: ", I + 1, ": ";
   leia nomeprog[I] ;
   imprima "\nMATRICULA: ";
   leia vetprog[I] ;
 }
 imprima "\n\nEntre com os dados dos alunos de Sistema de Informacao";
 para( I <- 0; I <= 99; I++)
```

```
{  imprima "\n NOME: ", I, ": ";
  leia nomesi [I] ;
  imprima "\nMATRICULA: ";
  leia vetsi[I] ;
}
faca
{
 imprima "\n\nLISTA";
 imprima "\n1 - LISTA TODOS DE  PROGRAMACAO I";
 imprima "\n2 - LISTA TODOS DE SISTEMAS DE INFORMACAO";
 imprima "\n3 - LISTA TODOS QUE FAZEM AS DUAS ";
 imprima "\n4 - SAIR ";
 imprima "\nOPCAO: ";
 leia OP;
 se ( OP == 1)
 {
   imprima "\n\n RELACAO DOS ALUNOS DE PROGRAMACAO  I\n";
   para( I <- 0; I <= 99; I++)
  {    imprima "\n",vetprog[ I] , "-", nomeprog[I] ; }
 }
 senao
 { se ( OP == 2)
   { imprima "\n\n RELACAO DOS ALUNOS  DE SISTEMA DE INFORMACAO \n";
     para( I <- 0; I <= 99; I++)
     {    imprima "\n",vetsi[ I] , "-", nomesi[I] ; }
   }
   senao
   { se ( OP == 3)
     { imprima "\n\n RELAÇO DOS ALUNOS QUE CURSAM SI / PROG. I \n";
       para( I <- 0; I <= 99; I++)
       {  J <- 0;
          enquanto(  J < 99 && vetprog[I] < > vetsi[J] )
          { J++; }
          se( vetprog[I] == vetsi[J] )
          { imprima "\n",vetsi[I] , "-", nomesi[I] ; }
       }
     }
     senao
     { se ( OP == 4)
       { imprima "\nFECHANDO";}
       senao
      {  imprima "\nOpcao nao disponivel";}
 } } }
 imprima "\n\n";
}
enquanto(  OP < > 4)
imprima "\n";
fimprog
```

algoritmo 397

Uma costureira tem dez freguesas fixas. Ela gostaria de fazer um algoritmo que funcionasse de acordo com o menu a seguir:

↳ *A costureira só faz vestidos (simples, passeio ou longo), calça, saia, blusa, conjunto e blazer.*

> *Atelier Maravilha*
>
> *1 – Cadastar as freguesas*
> *2 – Cadastrar preços das costuras*
> *3 - Calcular e imprimir o total que será pago por cada freguesa*
> *4 - Listar dados de uma cliente*
> *5 - Sair do programa*

```
prog vetor45
 int L, flag, flag1, op, ped,n ;
 string resp, nome[10], tel[10];
 real preco[8], soma,valor, pedido[10];
 para( L<- 0; L < 10 ; L++)
 { pedido[L] <- 0.;}
 flag <-0;
 flag1 <-0;
 faca
 {
 imprima "\n\n\n";
 imprima "\n Atelier Maravilha \n";
 imprima "\n1 - Cadastar as freguesas ";
 imprima "\n2 - Cadastrar precos das costuras ";
 imprima "\n3-  Calcular e imprimir o total que ser  pago por cada freguesa ";
 imprima "\n4-  Lista dados de uma cliente ";
 imprima "\n5 - Sai do programa ";
 imprima "\nOPCAO:";
 leia op;
 se( op == 1)
 { flag<-1;
   para( L<- 0; L < 10 ; L++)
  { imprima "\n nome " , L + 1, ": ";  leia nome[L] ;
   imprima "\n telefone " , L + 1, ": "; leia tel[L] ;
  }
 }
 senao
 { se(op == 2)
   {imprima "\n1 Ves-s 2 Ves-p 3 Ves-L  4 Conjunto 5 Blazer 6 Saia 7 Calca 8 blusa";
     para( L<- 0; L < 8; L++)
     { imprima "\n preco do  ", L + 1, ": "; leia preco[L];}
```

```
  flag1 <-1 ;
}
senao
{se(op == 3)
 { se(flag1 == 0)
   {imprima "\n Nao tem precos cadastrados";}
   senao
   {soma  <- 0.;
    imprima "\n1 Ves-s 2 Ves-p 3 Ves-L  4 Conjunto 5 Blazer 6 Saia 7 Calca 8
    blusa 0 para acabar";
    leia  ped;
    enquanto( ped < > 0)
    { enquanto(  ped < 1 ||  ped >8)
      {imprima "\n1 Ves-s 2 Ves-p 3 Ves-L 4 Conjunto 5 Blazer 6 Saia 7 Calca 8
      blusa r";
       leia  ped; }
      soma  <-  soma + preco[ped -1];
      imprima "\n1 Ves-s 2 Ves-p 3 Ves-L  4 Conjunto 5 Blazer 6 Saia 7 Calca 8
      blusa 0 para acabar";
      leia  ped; }
    imprima "\nTotal: ", soma;
    imprima  "\n Cliente cadastrada(S/N)? "; leia resp;
    se( resp == "S" || resp =="s")
    { para( L<-0; L < 10 ; L++)
      { imprima "\n " , L + 1, ": ",nome[L] ; }
      imprima  "\n Numero da cliente: "; leia n;
      enquanto(  n< 1 ||  n>10)
      { imprima "\nNumero de 1 - 10: "; leia  n; }
      pedido[n-1] <- soma;
    }
   }
 }
 senao
  { se(op==4)
    {
     para( L<-0; L < 10 ; L++)
     { imprima "\n " , L + 1, ": ",nome[L] ; }
     imprima  "\n Numero da cliente: ";
     leia n;
     enquanto(  n< 1 ||  n>10)
     {imprima "\nNumero de 1 - 10: "; leia  n; }
     imprima "\nSaldo R$: ", pedido[n -1];
     imprima  "\n Fazer pagamento(S / N)? ";
     leia resp;
     se( resp == "S" || resp =="s")
     {imprima  "\n Valor do pagamento: ";
      leia valor;
      pedido[n-1] <- pedido[n-1]- valor; }
    }
    senao
    { se(op==5)
```

```
        {imprima "\nSaindo do algoritmo"; }
        senao
        {imprima "\nOpcao nao disponivel"; }
      }
    }
   }
  }
 }
 enquanto(op< >5)
 imprima "\n";
fimprog
```

algoritmo 398

A Fábrica de Queijo Rio Novense deseja elaborar um algoritmo para controlar o estoque e as vendas. Inicialmente, deverão ser lidos e armazenados em vetores: o código, a quantidade disponível em estoque e o preço de venda dos produtos. O término do cadastramento é determinado quando se digita –1 para o código do produto. Sabe-se que a Fábrica de Queijo Rio Novense trabalha com no máximo 50 produtos diferentes.

A segunda fase do algoritmo é a venda. Deverá ser lido o código do produto a ser vendido e a quantidade requerida. Se o código do produto estiver cadastrado, a venda poderá ser realizada; caso contrário, a mensagem Produto Não-Cadastrado deverá ser exibida no monitor. Caso o produto esteja disponível, a venda só poderá ser realizada se a quantidade disponível no estoque for suficiente para atender ao pedido. Nesse caso, você deverá abater do estoque a quantidade vendida. Se o estoque não for suficiente para atender ao pedido, a mensagem Estoque Insuficiente deverá ser exibida no monitor. O final das vendas será detectado quando o código do produto for igual a zero.

No final, deverá aparecer uma listagem no vídeo contendo o total vendido no dia e a relação de todos os produtos do estoque, com suas respectivas quantidades, em ordem decrescente de quantidades disponíveis.

```
prog vetor46
 int k,codi,cod[50],quant[50],codigo,t,i,quantidade,aux;
 real preco[50], venda, geral, auxp;
 k<-0;
 geral <- 0.;
 imprima "\nDigite o codigo do produto ou 0 para acabar: ";
 leia codi;
 enquanto(k <50 && codi < > 0)
 { cod[k] <- codi;
  imprima "\nDigite a quantidade do produto: ";
  leia quant[k];
  imprima "\nDigite o preco do produto: ";
  leia preco[k];
  k++;
  imprima "\nDigite o codigo do produto ou 0 para acabar: ";
```

```
      senao
      { imprima "\nCentral completa"; }
   }
   senao
   { se(op==2)
     {  i <- 0;
        imprima "\nDigite o nome: ";
        leia nnome;
        enquanto(nnome < > nome[i] && i < k -1)
        { i++; }
          se(nnome == nome[i])
          {
            imprima "\nDigite o novo telefone: ";
            leia ntel;
            tel[i] <- ntel;
          }
          senao
          {  imprima "\nNome nao cadastrado"; }
     }
     senao
     {  se(op == 3)
        { k--;
          i <- 0;
          imprima "\nDigite o nome: ";
          leia nnome;
          enquanto(nnome < > nome[i] && i < k )
          { i++; }
          se(nnome == nome[i])
          { nome[i] <- "Vazio";   tel[i] <- 0;}
          senao
          {imprima "\nNome nao cadastrado"; }
        }
        senao
        { se(op==4)
          { para(i <- 0; i <= k -2; i++)
            {  para(cont <- i +1 ; cont <= k -1; cont++)
               { se(nome[i] > nome[cont])
                 {  auxnome <- nome[i];  nome[i] <- nome[cont];
nome[cont] <- auxnome;
                    auxtel  <- tel[i];   tel[i]<-tel[cont];  tel[cont] <-
auxtel;
                 }
               }
            }
            imprima "\nRelacao dos telefones";
            para(i <- 0; i < 1000 ; i++)
            {   se(nome[i] < > "Vazio")
                {
```

```
                imprima "\nNome: ", nome[i];
                imprima "\nTel: ", tel[i];
                imprima "\n";}
        }       }
        senao
        { se(op == 5)
          { i <- 0;
            imprima "\nDigite o nome: ";
            leia nnome;
            enquanto(nnome < > nome[i] && i < k)
            { i++; }
            se(nnome == nome[i])
            { imprima "\nNome: ", nnome;  imprima "\nTel: ", tel[i]; }
            senao
            {imprima "\nNome nao cadastrado"; }
          }
          senao
          { se ( op == 6)
            { imprima "\nSaindo";}
            senao
            {  imprima "\nOpcao nao disponivel";}
          } } } } }
   imprima "\n\n";
  }
  enquanto(  op < > 6)
  imprima "\n";
fimprog
```

algoritmo 400

Um hotel-fazenda gostaria de fazer um algoritmo que pudesse controlar os seguintes dados dos 50 quartos:

- *número de leitos por quarto;*
- *preço;*
- *situação: alugado, livre ou reservado;*
- *aluguel do quarto com data de entrada e de saída e número de diárias;*
- *despesas dentro do hotel;*
- *valor a ser pago;*
- *impressão de todos os quartos com a situação de cada um;*
- *impressão dos quartos livres.*

Criar um algoritmo que funcione de acordo com o menu a seguir:

Hotel-Fazenda Sucesso

1. Cadastra quartos

2. Lista todos os quartos

3. Lista quartos ocupados

4. Aluguel/reserva quarto

5. Entra despesas extras

6. Calcula despesa do quarto

7. Sai

```
prog vetor48
 int leitos[50], nd[50], i, k, chave, quarto, op;
 string sit[50], din[50], dout[50], nome[50], tel[50],resp, resp1,respo,
 nomep;
 real precos[50], despesas[50], valor, total;
 chave <- 0;
 faca
 {
   imprima "\n\n\nHotel Fazenda Sucesso \n";
   imprima "\n1. Cadastra quartos ";
   imprima "\n2. Lista todos os quartos ";
   imprima "\n3. Lista quartos desocupados ";
   imprima "\n4. Aluguel / reserva quarto ";
   imprima "\n5. Entra despesas extras ";
   imprima "\n6. Calcula despesa do quarto ";
   imprima "\n7. Sai";
   imprima "\nOpcao: ";
   leia op;
   imprima "\n";
   se(op==1)
   { se( chave ==1)
     { imprima "\nAtencao. Dados ja cadastrados"; }
     senao
     { para( i <- 0; i < 50; i++)
       # use este trecho se vc desejar ver o funcionamento
       { sit[i] <-"L"; nd[i] <- 0; despesas[i] <- 0.;
         din[i] <-"XXXX"; dout[i] <-"XXXX"; nome[i] <- "";
         tel[i] <- "";
         precos[i] <- 30.; leitos[i] <-5;}

         # o trecho abaixo e que seria o correto
       /*{ imprima "\nquantidade de leitos para o quarto ", i + 1, ": ";
         leia  leitos[i];
         imprima "\npreco do quarto: ";
```

```
        leia  precos[i];
        sit[i] <-"L";
        nd[i] <- 0;
        despesas[i] <- 0.;
        din[i] <-" XXXX";
        dout[i] <-" XXXX";
        nome[i] <- "";
        tel[i] <- "";
       } */

        chave <- 1;
     }
  }
  senao
  { se(op==2)
    { se( chave == 0)
      { imprima "\nEscolha a opcao 1"; }
      senao
      { para( i <- 0; i < 50; i++)
        { imprima "\nnumero do quarto: ", i +1, "\n";
          imprima "\nsituacao quarto: ", sit[i];
          se(sit[i]== "A" || sit[i] == "R")
          { imprima "\nnome: ", nome[i]; imprima "\ntelefone: ", tel[i];}
          imprima "\nquantidade de leitos: ", leitos[i];
          imprima "\npreco do quarto: ", precos[i];
          imprima "\ndespesas: ", despesas[i];
          imprima "\ndata de entrada: ", din[i];
          imprima "\ndata de saida: ", dout[i];
          imprima "\nnumero de dias: ", nd[i];
          imprima "\npressione enter para continuar: ";
          leia  respo;
        }
      }
    }
    senao
    { se(op==3)
      { se( chave ==0)
        { imprima "\nEscolha a opcao 1"; }
        senao
        { para( i <- 0; i < 50; i++)
          { se( sit[i] == "A")
           { imprima "\nnumero do quarto: ", i +1, "\n";
             imprima "\ndespesas: ", despesas[i];
             imprima "\ndata de entrada: ", din[i];
             imprima "\ndata de saida: ", dout[i];
             imprima "\nnumero de dias: ", nd[i];
             imprima "\npressione enter para continuar: ";
             leia  resp;
```

```
        }
      }
    }
  }
  senao
  { se(op==4)
    { se( chave ==0)
      { imprima "\nEscolha a opcao 1"; }
      senao
      { imprima "\nDigite A( aluguel)  R(reserva): ";
        leia  resp;
        enquanto( resp < > "A" && resp < > "a" && resp < > "R" &&
        resp < > "r")
        { imprima "\nResposta invalida. Digite ( aluguel)
        R(reserva): ";
          leia  resp; }
        se( resp == "A"  || resp == "a")
        { imprima "\n Tinha reserva ( s /n)? ";leia  resp1;
          se( resp1 == "S"  || resp1 == "s")
          { imprima "\nEntre com  nome: "; leia  nomep;
            i <- 0;
            enquanto( nomep < > nome[i] && i < 49 )
            { i++;}
            se(nomep == nome[i] )
            { quarto <- i; sit[i] <- "A"; despesas [i] <- 0.;
              imprima "\ndata de entrada: ";leia din[quarto];
              imprima "\ndata de saida: "; leia dout[quarto];
              imprima "\nnumero de dias: ";leia  nd[quarto];
            }
            senao
            { imprima "\nReserva nao encontrada ";
              resp1  <- "N"; }
            }
        }
        se(resp1 == "N" || resp1 == "n" || resp=="r" || resp=="R")
        { para( i <- 0; i < 50; i++)
          { se( sit[i] == "L")
            { imprima "\nnumero do quarto: ", i +1, "\n";
              imprima "\nquantidade de leitos: ", leitos[i];
              imprima "\npreco do quarto: ", precos[i];
              imprima "\npressione enter para continuar: ";
              leia  respo;
            }
         }
         imprima "\nDigite o numero do quarto para aluguel ou
         reserva: ";
         leia  quarto;
         enquanto( quarto < 1|| quarto > 50)
```

```
   { imprima "\nQuarto Invalido. Entre novamente: ";
     leia  quarto; }
    enquanto(sit[quarto-1]  == "A")
   { imprima "\nQuarto ocupado. Digite novamente: ";
     leia  quarto; }
     se( resp == "A"  || resp == "a")
    { sit[quarto-1] <- "A"; despesas [quarto -1] <- 0.;}
   senao
   { sit[ quarto -1]  <- "R"; }
   imprima "\nDigite nome: "; leia nome[quarto -1];
   imprima "\nDigite telefone para contato: "; leia
   tel[quarto-1];
   imprima "\ndata de entrada: "; leia din[quarto -1];
   imprima "\ndata de saida: ";  leia dout[quarto -1];
   imprima "\nnumero de dias: ";leia  nd[quarto-1];
  }
 } }
 senao
 { se(op==5)
   { se( chave ==0)
     { imprima "\nEscolha a opcao 1"; }
     senao
     {  imprima "\nentre com numero do quarto: ";
        leia  quarto;
        enquanto( quarto < 1|| quarto > 50)
        { imprima "\nentre com numero do quarto: ";
          leia  quarto; }
        imprima "\ndespesas: ";
        leia valor;
        despesas[quarto-1]  <- despesas[quarto-1]+ valor;
     }
   }
   senao
   { se(op==6)
     { se( chave ==0)
       { imprima "\nEscolha a opcao 1"; }
     senao
     { imprima "\nentre com numero do quarto: ";
       leia  quarto;
       enquanto( quarto < 1|| quarto > 50)
       { imprima "\nentre com numero do quarto: ";
         leia  quarto; }
       total <- despesas[quarto-1] + precos[quarto-1];
       imprima "\ndespesas: R$  ", total;
       sit[quarto-1] <-"L";
       despesas[quarto-1] <- 0.;
       din[quarto-1] <- "XXXX";
       dout[quarto-1] <-"XXXX";
```

```
                    nd[quarto-1] <- 0;
                    nome[quarto-1] <-"";
                  }
                }
                senao
                { se(op==7)
                  { imprima "\nSaindo ";}
                  senao
                  {imprima "\nOpcao nao disponivel";}
 } } } } } }
 }
 enquanto( op< >7)
 imprima "\n\n";
fimprog
```

algoritmo 401

Criar um algoritmo que funcione de acordo com o menu a seguir, sabendo-se que poderão ser cadastradas até 50 pessoas.

```
********************
*        MENU       *
********************
1 - cadastra dados do cliente
2 - cadastra milhagem do cliente
3 - lista milhagem do cliente
4 - imprime os nomes que têm maior e menor milhagem
5 - imprime os nomes e as milhagens
6 - SAIR
OPCAO:
```

OBSERVAÇÃO

*A maioria das soluções foi feita com **se**s aninhados para que você pudesse testar no interpretador, mas muitas delas teriam melhor visualização se fossem feitas com escolha. Terminaremos essa lista com um execício com as duas soluções.*

```
prog vetor49
 #  nao roda na versao 2 do UAL, troque escolha por ses aninhados para
 ver a execucao
 int op,c,d, posmenor, posmaior;
 string  dados1[50], dados2[50], dados3[50], nome;
 real    milha[50]. milhas;
 para( c<-0;c < 5; c++)
 {milha[c]<-0.;}
 c<-0;
 faca
```

```
{ imprima "\n\n\n*******************";
  imprima "\n   *       MENU               *";
  imprima "\n   *******************";
  imprima "\n 1 - cadastra dados do cliente   ";
  imprima "\n 2 - cadastra milhagem do cliente ";
  imprima "\n 3 - lista milhagem do cliente    ";
  imprima "\n 4 - imprime os nomes que tem maior e menor milhagem ";
  imprima "\n 5 - imprime os nomes e as milhagens ";
  imprima "\n 6 - SAIR           ";
  imprima "\nOPCAO:";
  leia op;
   escolha(op)
   {
      caso 1:
         se(c<50)
         {
           imprima "\n", c+ 1,"- nome: ";leia dados1[c];
           imprima "\nendereco: "; leia dados2[c];
           imprima "\ntelefone:"; leia dados3[c];
           c++;
           }
          senao
         { imprima "\narquivo cheio";}
         pare;
      caso 2:
        imprima "\nnome para procura: ";leia nome;
        d<-0;
        enquanto( d< c - 1  && nome < > dados1[d]  )
        { d++; }
        se(nome==dados1[d]  )
        {  imprima  "\ndigite milhagem de ", dados1[d] ,": ";leia milhas;
           milha[d] <-milha[d] +milhas;
        }
       senao
        { imprima "\nnome nao encontrado";}
        pare;
    caso 3:
        imprima "\nnome para procura: ";leia nome;
        d<-0;
        enquanto(d< c - 1  && nome < > dados1[d]  )
        { d++; }
        se(nome==dados1[d]  )
        {  imprima  "\nmilhagem  de ",dados1[d] ,": ",milha[d] ;
        senao
        { imprima "\nnao encontrado";}
        pare;
    caso 4:
      d<-1;
      posmenor<-0;
      posmaior<-0;
```

```
        enquanto(d <= c)
        { se(milha[d]  > milha[posmaior])
        { posmaior<-d;}
         senao
         { se(milha[d] <milha[posmenor])
           { posmenor<-d; }
         }
        d++;
       }
       imprima "\nDados da pessoa de maior milhagem";
       imprima "\nNome: ",dados1[posmaior];
       imprima "\nEndereco: ",dados2[posmaior];
       imprima "\nTelefone: ",dados3[posmaior];
       imprima "\nMilhagem: ",milha[posmaior];
       imprima "\n\n";
       imprima "\nDados da pessoa de menor milhagem";
       imprima "\nNome: ",dados1[posmenor];
       imprima "\nEndereco: ",dados2[posmenor];
       imprima "\nTelefone: ",dados3[posmenor];
       imprima "\nMilhagem: ",milha[posmenor];
       pare;
     caso 5:
       imprima "\nListagem";
       para( d<-0;d< c; d++)
       { imprima "\n", d,"- ",dados1[d] ,": ",milha[d] ;}
       pare;
     caso 6:
      imprima "\nBOA VIAGEM";
      pare;
     senao
      imprima "\nOpcao inexistente ";
   }
  }
  enquanto(  op < >  6)
  imprima "\n";
 fimprog

 prog vetor49
  int op,c,d, posmenor, posmaior;
  string   dados1[50], dados2[50], dados3[50], nome;
  real    milha[50], milhas;
  para( c<-0;c < 5; c++)
  {milha[c]<-0.;}
  c<-0;
  faca
  { imprima "\n\n\n*******************";
    imprima "\n   *       MENU               *";
    imprima "\n   *******************";
    imprima "\n 1 - cadastra dados do cliente   ";
    imprima "\n 2 - cadastra milhagem do cliente ";
```

```
imprima "\n 3 - lista milhagem do cliente     ";
imprima "\n 4 - imprime os nomes que tem maior e menor milhagem ";
imprima "\n 5 - imprime os nomes e as milhagens ";
imprima "\n 6 - SAIR ";
imprima "\nOPCAO:";
leia op;
 se(op==1)
 { se(c<=49)
   {
     imprima "\n", c+ 1,"- nome: ";leia dados1[c];
     imprima "\nendereco: "; leia dados2[c];
     imprima "\ntelefone:"; leia dados3[c];
     c++;
   }
   senao
   { imprima "\narquivo cheio";}
  }
  senao
  {se( op ==2)
  { imprima "\nnome para procura: ";leia nome;
    d<-0;
    enquanto( d< c-1  && nome < > dados1[d]  )
    { d++; }
    se(nome==dados1[d]  )
    {  imprima  "\ndigite milhagem de ", dados1[d] ,": ";leia milhas;
       milha[d] <-milha[d] +milhas;
    }
    senao
    { imprima "\nnome nao encontrado";}
  }
  senao
  { se( op == 3)
   { imprima "\nnome para procura: ";leia nome;
       d<-0;
       enquanto( d< c-1  && nome < > dados1[d]  )
       { d++; }
       se(nome==dados1[d]  )
       {  imprima  "\nmilhagem  de ",dados1[d] ,": ",milha[d] ; }
       senao
       { imprima "\nnao encontrado";}
  }
  senao
  { se(  op == 4)
    {  d <-1;
       posmenor<-0;
       posmaior<-0;
       enquanto(d < c)
       { se(milha[d]  > milha[posmaior])
         { posmaior <- d;}
         senao
```

```
            { se(milha[d] < milha[posmenor])
              { posmenor <- d; }
             }
             d++;
          }
          imprima "\nDados da pessoa de maior milhagem";
          imprima "\nNome: ",dados1[posmaior];
          imprima "\nEndereco: ",dados2[posmaior];
          imprima "\nTelefone: ",dados3[posmaior];
          imprima "\nMilhagem: ",milha[posmaior];
          imprima "\n\n";
          imprima "\nDados da pessoa de menor milhagem";
          imprima "\nNome: ",dados1[posmenor];
          imprima "\nEndereco: ",dados2[posmenor];
          imprima "\nTelefone: ",dados3[posmenor];
          imprima "\nMilhagem: ",milha[posmenor];
        }
        senao
        { se( op == 5)
          { imprima "\nListagem";
            para( d<-0;d< c; d++)
            { imprima "\n", d+1,"- ",dados1[d] ,": ",milha[d] ;}
          }
          senao
          { se( op == 6)
            { imprima "\nBOA VIAGEM";}
            senao
            { imprima "\nOpcao inexistente ";}
  } } } } }
 }
 enquanto(  op < >  6)
 imprima "\n";
fimprog
```

CONCEITOS DE MATRIZES

Nossa experiência mostra que este é um dos momentos críticos no aprendizado de algoritmos e vamos tentar ajudá-lo, recordando alguns conceitos matemáticos. Uma matriz, na verdade, é uma tabela contendo elementos que servirão de base para vários cálculos em muitas ciências tais como Estatística, Economia, Física, Ciência da Computação etc.

Observando os exemplos a seguir, perceberemos que os elementos estão dispostos em linhas e colunas:

$$A\begin{bmatrix} 2 & 3 \\ 6 & 8 \end{bmatrix} \qquad D\begin{bmatrix} 3 & 12 \\ 14 & 8 \\ 15 & 17 \end{bmatrix} \qquad X\begin{bmatrix} 2 & 4 & 5 \\ 6 & 9 & 10 \\ 3 & 5 & 8 \end{bmatrix}$$

A 2 linhas e 2 colunas   *D 3 linhas e 2 colunas*   *X 3 linhas e 3 colunas*

$$B\begin{bmatrix} 6 & 8.7 & 15 \\ 3.6 & 4 & 12 \end{bmatrix}$$

B 2 linhas e 3 colunas

$$M\begin{bmatrix} 7 & 8 & 8 & 5.3 & 7 & 9.3 \\ 9 & 10 & 6.5 & 2.3 & 8 & 10 \\ 7.2 & 8.7 & 10 & 7.8 & 3.4 & 9 \end{bmatrix}$$

M 3 linhas e 6 colunas

$$Q\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Q 4 linhas e 4 colunas

Se considerarmos m como sendo o número de linhas e n o número de colunas, poderemos afirmar que:

- a matriz A é de ordem 2 × 2
- a matriz B é de ordem 2 × 3
- a matriz D é de ordem 3 × 4
- a matriz M é de ordem 3 × 6
- a matriz X é de ordem 3 × 3
- a matriz B é de ordem 4 × 4

Quando **m** (número de linhas) é diferente de **n** (número de colunas), dizemos que estas matrizes são retangulares.

Quando **m** (número de linhas) é igual a **n** (número de colunas), dizemos que estas matrizes são quadradas.

Será que o fato de ser retangular ou quadrada tem alguma diferença no estudo de matrizes em Computação?

Se considerarmos a forma como serão dimensionadas, não; mas se os algoritmos envolverem conteúdos matemáticos, sim, pois os matemáticos têm estudado a Álgebra das matrizes por muitos anos e já desenvolveram várias propriedades que são peculiares às matrizes quadradas. Mas isso veremos nos exemplos de algoritmos que envolvam matrizes.

Algo que nos preocupa é o fato de os alunos conseguirem lidar bem com vetores e terem muita dificuldade com matrizes. Você sabia que uma matriz de ordem **m** por **1** (**n** = **1**) é chamada de matriz-coluna ou vetor-coluna e uma matriz de ordem **1** (**m** = **1**) por **n** é chamada de matriz-linha ou vetor-linha?

Então, vamos desmistificar o estudo de matrizes em programação, pois, na verdade, se agruparmos vários vetores, teremos uma matriz.

Vamos agora mostrar que você sempre esteve familiarizado com matrizes que, muitas vezes, chamava de tabelas:

1. Tabela contendo suas notas nas disciplinas que você está cursando

Disciplina	PR1	PR2	MÉDIA	P. FINAL	MÉDIA
Programação I					
Sistemas de Informação					
Matemática					

2. Tabela contendo suas despesas

Despesa	Janeiro	Fevereiro	Março
Faculdade	400,00	400,00	420,00
Passagem	50,00	50,00	50,00
Prestação do carro	320,00	320,00	320,00
Alimentação	50,00	120,00	150,00

3. Tabela contendo dados dos funcionários

Nome	Profissão	Cargo	Local
ANITA LUIZA MACIEL LOPES	Professora	Professor	UNESA
CARLOS AUGUSTO GARCIA ASSIS	Professor	Coordenador	UNESA

Você agora deverá estar pensando: mas eu não posso fazer tudo isso com vetores?

Muitas vezes será possível, mas, em outros casos, seu algoritmo ficará muito extenso ou impossível de resolver se for de base matemática.

Dimensionando uma matriz

Sabendo-se que para dimensionar uma matriz usamos o seguinte comando na declaração de variáveis:

```
tipo [dimensão1] [dimensão2]
```

onde dimensão, na prática, significa o intervalo do número de linhas, colunas etc.

[5] [10] 5 linhas e 10 colunas

tipo poderá ser: **inteiro, real** ou **string**

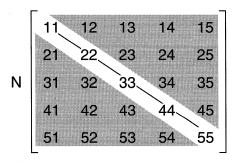
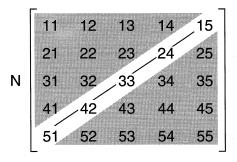
nome será o que você dará à matriz dentro das regras para se nomear uma variável.

Exemplos:
Dimensionando as matrizes dos exemplos iniciais A, B, M e X:

int A[2][2]; **real** M[3][6];

real B[2][3]; **int** X[3][3];

Algumas considerações

Vamos observar as posições de uma matriz de ordem 5 (quando falamos assim, estamos nos referindo a uma matriz quadrada):

1. Os elementos que se encontram nas posições 11, 22, 33, 44 e 55 formam a **Diagonal Principal.** Observe que o número da linha é sempre igual ao da coluna.
2. Os elementos que se encontram nas posições 12, 13, 14, 15, 23, 24, 25, 34, 35 e 45 formam o **triângulo superior.** Observe que o número da linha é sempre menor do que o da coluna.
3. Os elementos que se encontram nas posições 21, 31, 32, 41, 42, 43, 51, 52, 53 e 54 formam o **triângulo inferior.** Observe que o número da linha é sempre maior do que o da coluna.

4. Os elementos que se encontram nas posições 15, 24, 33, 42 e 51 formam a **Diagonal Secundária.** Observe que o número da linha somado ao número da coluna é sempre igual a 6, isto é, **ordem + 1.**

5. Os elementos que se encontram nas posições 11, 12, 13, 14, 21, 22, 23, 31, 32 e 41 formam o **triângulo superior.** Observe que o número da linha somado ao número da coluna é sempre menor ou igual a **ordem.**
6. Os elementos que se encontram nas posições 25, 34, 35, 43, 44, 45, 52, 53, 54 e 55 formam o **triângulo inferior.** Observe que o número da linha somado ao número da coluna é sempre maior ou igual a **ordem + 2.**

DIAGONAL PRINCIPAL	DIAGONAL SECUNDÁRIA
elementos da DP: L = C	elementos da DS: L + C = ordem + 1
elementos acima da DP: L < C	elementos acima da DS: L + C< = ordem
elementos abaixo da DP: L > C	elementos abaixo da DS: L + C > = ordem + 2

Para que possamos imprimir os elementos de uma matriz, precisaremos usar um Para dentro de outro Para. Vamos recordar um trecho que vimos no capítulo anterior quando desejávamos imprimir todos os arranjos possíveis com números:

Recordando o trecho de ARRANJO:

```
para( L<- 1;  L <= ordem ; L++)
{ para( C<- 1;  c <= ordem ; c++)
  {   se (L < > C)
      {    imprima "\n",L," - ", C; }
   }
}
```

Adaptando para MATRIZ

```
para( L<- _____;  L <= _____ ; L++)
{
   para( C<- _____;  C <= _____; C++)
  {
       se (expressão)
       { imprima nomematriz[L] [ c], "ƀƀ"; }
  }
  imprima "\n";
}
Observação 1: expressão é uma das citadas anteriormente
```

Essa solução não imprime a matriz como na matemática. Mais adiante, incluindo o caractere de controle \t, mostraremos como fazer isso.

Por que esta solução é dita ineficiente?

Porque são feitos muitos testes. Exemplo: Suponha uma matriz de ordem 60:

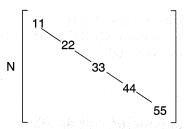
- 60 × 60 = 3600 elementos, logo 3600 testes
- retirando os elementos que se encontram na diagonal, teremos: 3600 – 60 =**3540**
- dividindo por dois, teremos **1770** testes necessários se desejarmos só os elementos abaixo ou acima da diagonal
- **60** testes necessários se forem os da diagonal.

Qual seria a melhor solução?

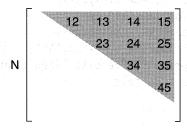
Seria *triangular a matriz*, pois retiraríamos todos os testes (estrutura do se). Observe os trechos para impressão de uma matriz N a seguir:

TRIANGULAÇÃO DE MATRIZES

Diagonal Principal:

```
...
 para(L <- 1; L<= 5;L++)
{  imprima N[L][L],"\n";
  para(t <- 1;t <= L; t++)
  {   imprima "\t"; }
 }
...
```

*Como a linha é igual à coluna, não precisamos de dois **paras**.*

Triângulo superior da DP:

```
...
 para( L <- 1; L <= 4;L++)
 {
  para(C <- L+1; C<= 5; C++)
   {    imprima "\t",N[L][C];    }
   imprima "\n";
   para(t <- 1; t <= L; t++)
   {imprima "\t";}
 }
 imprima "\n";
...
```

Triângulo inferior da DP:

```
...
 para(L <- 2; L <= 5;L++)
 {  imprima "\n";
  para(C <- 1; C < L; C++)
   {   imprima N[L][C],"\t"; }
 }
...
```

Diagonal Secundária:

```
...
 para(L<-1;L<=5;L++)
 {
  para(t<-5-L;t>=1;t--)
  {   imprima "\t";  }
   imprima N[L][6-L],"\n";
 }
...
```

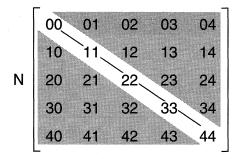
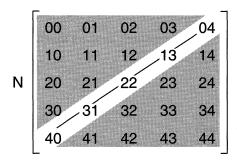
Triângulo superior da DS:

```
...
 para(L<-1;L<=4;L++)
 {
  para(c<-1;c<6-L;c++)
  {   imprima N[L[[c],"\t";    }
   imprima "\n";
 }
...
```

Triângulo inferior da DS:

```
...
 para(L<-2;L<=5;L++)
 {
  para(t<-6-L;t>=1;t--)
  {imprima "\t"; }
  para(c<-7-L;c<=5;c++)
  { imprima L, "-",c,"\t";    }
 imprima "\n";# esta linha podera  nao ser necessária se voce usar as 10
 zonas da tela
 }
...
```

Como já vimos anteriormente, muitas linguagens trabalham com as posições das matrizes, começando na linha 0 e na coluna 0.

Vamos observar as posições de uma matriz de ordem 5:

1. Os elementos que se encontram nas posições 00, 11, 22, 33 e 44 formam a **Diagonal Principal.** Observe que o número da linha é sempre igual ao da coluna.
2. Os elementos que se encontram nas posições 01, 02, 03, 04, 12, 13, 14, 23, 24 e 34 formam o **triângulo superior.** Observe que o número da linha é sempre menor do que o da coluna.
3. Os elementos que se encontram nas posições 10, 20, 21, 30, 31, 32, 40, 41, 41, 42 e 43 formam o **triângulo inferior.** Observe que o número da linha é sempre maior do que o da coluna.

4. Os elementos que se encontram nas posições 04, 13, 22, 31 e 40 formam a **Diagonal Secundária.** Observe que o número da linha somado ao número da coluna é sempre igual a 4, isto é, **ordem – 1.**
5. Os elementos que se encontram nas posições 00, 01, 02, 03, 10, 11, 12, 20, 21, e 30 formam o **triângulo superior.** Observe que o número da linha somado ao número da coluna é sempre menor ou igual a **ordem – 2.**
6. Os elementos que se encontram nas posições 14, 23, 24, 32, 33, 34, 41, 42, 43 e 44 formam o **triângulo inferior.** Observe que o número da linha somado ao número da coluna é sempre maior ou igual a **ordem.**

DIAGONAL PRINCIPAL	DIAGONAL SECUNDÁRIA
elementos da DP: L = C	elementos da DS: L + C = ordem – 1
elementos acima da DP: L < C	elementos acima da DS: L + C< = ordem – 2
elementos abaixo da DP: L > C	elementos abaixo da DS: L + C > = ordem

Veremos outras operações com matrizes e algumas considerações no momento oportuno.

EXERCÍCIOS – LISTA 6
LISTA DE MATRIZES

↳ *Assumiremos que as matrizes começam na linha 0 e coluna 0.*

algoritmo 402

Criar um algoritmo que leia os elementos de uma matriz inteira 10 × 10 e escreva os elementos da diagonal principal.

```
prog matriz1
 int N[10][10],L, c, t;
 para(L <-0; L <=9; L++)
 { para(c <-0; c <=9; c++)
  { imprima "\nEntre com elemento linhaƀ",L+1,"ƀƀcolunaƀ ",c+1, ":ƀ";
    leia N[L][c] ;
  }
 }
 imprima "\nDIAGONAL PRINCIPAL\n";
 para(L<-0;L<=9;L++)
 {  imprima  N[L][L], "\n";
  para(t <- 0;t <= L;t++)
   {   imprima  "\t";   }
 }
 imprima "\n";
fimprog
```

algoritmo 403

Criar um algoritmo que leia os elementos de uma matriz inteira 10 × 10 e escreva todos os elementos, exceto os elementos da diagonal principal.

```
prog matriz2
 int N[10][10],L, c;
 para(L <-0; L <=9; L++)
 { para(c <-0; c <=9; c++)
  { imprima "\nEntre com elemento linhaƀ", L + 1, "ƀƀcolunaƀ ",c  + 1,
":ƀ";
    leia N[L][c] ;
  }
 }
 imprima "\n\tTODOS EXCETO DP\n";
 para(L <- 0; L < 10; L++)
 {
 para(c <- 0; c < 10; c++)
 {
  se(L < > c)
   {  imprima N[L][c]; }
   imprima "\t ";
  }
    imprima "\n";
 }
 imprima "\n";
fimprog
```

algoritmo 404

Criar um algoritmo que leia os elementos de matriz inteira 10 × 10 e escreva somente os elementos acima da diagonal principal.

```
prog matriz3
 int N[10][10],L, c, t;
 para(L <-0; L <=9; L++)
 { para(c <-0; c <=9; c++)
  { imprima "\nEntre com elemento linhaƀ", L + 1, "ƀƀcolunaƀ ",c  + 1,
  ":ƀ";
    leia N[L][c] ;
  }
 }
 imprima "\nACIMA DA DIAGONAL PRINCIPAL\n";
 para(L<- 0;L <= 8;L++)
 {
  para(c <- L+1;c <= 9;c++)
   {   imprima  "\t", N[L][c];   }
  imprima "\n";
   para(t <- 0;t <= L;t++)
   {imprima  "\t";}
 }
 imprima "\n";
fimprog
```

algoritmo 405

Criar um algoritmo que leia os elementos de matriz inteira 10 × 10 e imprima a soma dos elementos que estão acima da diagonal principal.

```
prog matriz4
 int N[10][10],L, c, soma;
 soma  <- 0;
 para(L <-0; L <=9; L++)
 { para(c <-0; c <=9; c++)
  { imprima "\nEntre com elemento linhaƀ", L + 1, "ƀƀcolunaƀ ",c  + 1,
  ":ƀ";
    leia N[L][c] ;
  }
 }
 para(L<- 0;L <= 8;L++)
 {
 para(c <- L+1;c <= 9;c++)
   {  soma  <- soma  + N[L][c]; }
 }
 imprima "\n\tSoma dos elementos acima  da DP: ", soma;
 imprima "\n";
fimprog
```

algoritmo 406

Criar um algoritmo que leia os elementos de matriz inteira 10 × 10 e escreva somente os elementos abaixo da diagonal principal.

```
prog matriz5
 int N[10][10],L, c, t;
 para(L <-0; L <=9; L++)
 { para(c <-0; c <=9; c++)
  { imprima "\nEntre com elemento linhaƀ", L + 1, "ƀƀcolunaƀ ",c  + 1,
  ":ƀ";
    leia N[L][c] ;
  }
 }
 imprima "\nABAIXO DA DIAGONAL PRINCIPAL\n";
 para(L<-1;L<=9;L++)
 {   para(c <- 0;c < L;c++)
   {  imprima  N[L][c], "\t"; }
 }
 imprima "\n";
fimprog
```

algoritmo 407

Criar um algoritmo que leia os elementos de matriz inteira 10 × 10 e imprima o produto dos elementos que estão abaixo da diagonal principal.

```
prog matriz6
 int N[10][10],L, c, prod;
 prod  <- 1;
 para(L <-0; L <=9; L++)
 { para(c <-0; c <=9; c++)
  { imprima "\nEntre com elemento linhaƀ", L + 1, "ƀƀcolunaƀ ",c  + 1,
  ":ƀ";
    leia N[L][c] ;
  }
 }
 imprima "\nABAIXO DA DIAGONAL PRINCIPAL\n";
 para(L<-1;L<=9;L++)
 {
   para(c <- 0;c < L;c++)
   { prod <- prod  * N[L][c];    }
 }
 imprima "\n\tProdutorio dos elementos abaixo  da DP: ", prod;
 imprima "\n";
fimprog
```

algoritmo 408

Criar um algoritmo que leia os elementos de uma matriz inteira 10 × 10 e escreva os elementos da diagonal secundária.

```
prog matriz7
 int N[10][10],L, c, t;
 para(L <-0; L <=9; L++)
 { para(c <-0; c <=9; c++)
  { imprima "\nEntre com elemento linhaƀ", L + 1, "ƀƀcolunaƀ ",c  + 1,
  ":ƀ";
    leia N[L][c] ;
  }
 }
 imprima "\nDIAGONAL SECUNDARIA\n";
 para(L<-0;L<=9;L++)
 {
  para(t<-9-L;t>=1;t--)
  {   imprima   "\t";  }
  imprima  N[L][9-L], "\n";
   }
 imprima "\n";
fimprog
```

algoritmo 409

Criar um algoritmo que leia os elementos de matriz inteira 10 × 10 e escreva todos os elementos exceto os elementos da diagonal secundária.

```
prog matriz8
 int N[10][10],L, c;
 para(L <- 0; L < 10; L++)
 {
 para(c <- 0; c < 10; c++)
 {
 imprima "digite elemento de N[" ,L+1, "][",  c+1,  "]: ";
  leia N[L][c];
 }
 }
 imprima "\n\tTODOS EXCETO DS\n";
 para(L <- 0; L < 10; L++)
 {
  para(c <- 0; c < 10; c++)
  {
    se( L + c < > 9)
    { imprima N[L][c]; }
   imprima "\t";
  }
  imprima "\n";
 }
 imprima "\n";
fimprog
```

algoritmo 410

Criar um algoritmo que leia os elementos de matriz inteira 10 × 10 e escreva somente os elementos acima da diagonal secundária.

```
prog matriz9
 int N[10][10],L, c, t;
 para(L <-0; L <=9; L++)
 { para(c <-0; c <=9; c++)
  { imprima "\nEntre com elemento linhaƀ", L + 1, "ƀƀcolunaƀ ",c  + 1,
  ":ƀ";
    leia N[L][c] ;
  }
 }
 imprima "\nACIMA DA DIAGONAL SECUNDARIA\n";
 para(L <- 0;L <= 8;L++)
 {
   para(c <- 0;c<= 8-L;c++)
   {  imprima  N[L][c], "\t";  }
   imprima "\n";
```

```
}
imprima "\n";
fimprog
```

algoritmo 411

Criar um algoritmo que leia os elementos de matriz inteira 10 × 10 e escreva somente os elementos abaixo da diagonal secundária.

```
prog matriz10
 int N[10][10],L, c, t;
 para(L <- 0; L <= 9; L++)
 { para(c <- 0; c <= 9; c++)
  { imprima "\nEntre com elemento linhaƀ", L + 1, "ƀƀcolunaƀ ",c  + 1,
  ":ƀ";
    leia N[L][c] ;
  }
 }
 imprima "\nABAIXO DA DIAGONAL SECUNDARIA\n";
 imprima "\n";
 para(L<-1;L<=9;L++)
 {
  para(t<-9-L;t>=1;t--)
   {imprima  "\t"; }
   para(c<-10-L;c<=9;c++)
   {  imprima  N[L][c], "\t";  }
 }
 imprima "\n";
fimprog
```

algoritmo 412

Criar um algoritmo que armazene dados inteiros em uma matriz de ordem cinco e imprima: toda a matriz e uma outra matriz formada pelos números que se encontram em posições cuja linha mais coluna formam um número par.

```
prog matriz11
 int n[5][5],L, c, t;
 para(L<-0;L<5;L++)
 {  para(c<-0;c<5;c++)
    { imprima "\ndigite elemento:  ",L+1," - ",c+1," : ";leia n[L][c]; }
 }
 imprima "\nToda a matriz\n",;
 para(L<-0;L<5;L++)
 { para(c<-0;c<5;c++)
  { imprima n[L][c], "\t"; }
   imprima "\n";
 }
```

```
 imprima "\noSo os elementos nas posicoes cuja linha mais coluna resulta
 num par \n ";
 para(L<-0;L<5;L++)
 { para(c<-0;c<5;c++)
  { se( (L + c) % 2 ==0)
    { imprima n[L][c]; }
   imprima "\t";
  }
  imprima "\n";
 }
 imprima "\n";
fimprog
```

algoritmo 413

Criar um algoritmo que armazene dados inteiros em uma matriz de ordem cinco e imprima: toda a matriz e a raiz quadrada da soma dos quadrados dos números ímpares localizados abaixo da DS.

```
prog matriz12
 int n[5][5],L, c, t, soma;
 para(L<-0;L<5;L++)
 {  para(c<-0;c<5;c++)
   { imprima "\ndigite elemento:  ",L+1," - ",c+1," : ";leia n[L][c]; }
 }
 imprima "\nToda a matriz\n",;
 para(L<-0;L<5;L++)
 { para(c<-0;c<5;c++)
   { imprima n[L][c], "\t"; }
   imprima "\n";
 }
 soma<-0;
 para(L<-1;L<5;L++)
 {
   para(c<-5-L;c<5;c++)
   { se(n[L][c] % 2 == 1)
     { soma <- soma + n[L][c] * n[L][c];}
     }
 }
 imprima "\n\nraiz quadrada da soma dos quadrados dos numeros impares: ",
 sqrt(soma;
 imprima "\n";
fimprog
```

algoritmo 414

Criar um algoritmo que armazene dados em uma matriz de ordem 5 e imprima: toda a matriz e a matriz gerada só com números ímpares.

```
prog matriz13
 int n[5][5],L, c, t, soma;
 para(L<-0;L<5;L++)
 {  para(c<-0;c<5;c++)
   { imprima "\ndigite elemento:  ",L+1," - ",c+1," : ";leia n[L][c]; }
 }
 imprima "\nToda a matriz\n",;
 para(L<-0;L<5;L++)
 { para(c<-0;c<5;c++)
   { imprima n[L][c], "\t"; }
   imprima "\n";
 }
 imprima "\noSo os elementos impares\n ";
 para(L<-0;L<5;L++)
 { para(c<-0;c<5;c++)
 { se( n[L][c] % 2 ==1)
    { imprima n[L][c]; }
    imprima "\t";
 }
  imprima "\n";
 }
 imprima "\n";
fimprog
```

algoritmo 415

Criar um algoritmo que armazene nomes de cinco disciplinas, sabendo-se que as turmas têm sempre 15 alunos. Armazene também a matrícula e as notas dos alunos em matrizes de ordem cinco. Imprima a saída da seguinte maneira:

<Nome da disciplina>	
Mat.	*Nota*

```
prog matriz14
 int n[5][15],L, c, t;
 real notas[5][15];
 string dis[5], resp;
 para(L<-0;L<5;L++)
 {  imprima "\ndigite disciplina:  ",L+1, ":  ";leia dis[L];
  para(c<-0;c<15;c++)
  {  imprima \ndigite matricula do aluno: ",c+1, ":  "; leia n[L][c];
     imprima "\ndigite nota matricula do aluno: ",c+1, ":  ";
     leia notas[L][c];
  }
 }
 para(L<-0;L<5;L++)
 { imprima "\n\n", dis[L],"\n";
   para(c<-0;c<15;c++)
   { imprima "\n", n[L][c],"\t", notas[L][c], "\n"; }
    imprima "\npressione enter: ";# para pausar a tela
 }
 imprima "\n";
fimprog
```

algoritmo 416

Entrar com valores reais para uma matriz M[4] [5]. Gerar e imprimir a matriz METADE.

```
prog matriz15
 real M[4][5], METADE[4][5]; int L, c;
 para(L <- 0; L<4; L++)
 {
     para(c <- 0; c<5; c++)
     {
      imprima "digite elemento de M[" ,L+1, "][",  c+1,  "]: ";
      leia M[L][c];
      DOBRO[L][c] <- M[L][c] / 2;
     }
 }
 imprima "\n\t\tMATRIZ METADE\n";
 para(L <- 0; L<4; L++)
 {
  para(c <- 0; c<5; c++)
  {  imprima METADE[L][c], "\t";  }
  imprima "\n";
 }
 imprima "\n";
fimprog
```

algoritmo 417

Entrar com valores para uma matriz $A_{3 \times 4}$. Gerar e imprimir uma matriz B que é o triplo da matriz A.

```
prog matriz16
 int A[3][4], B[3][4], L,c;
 para(L <- 0; L<3; L++)
 {
  para(c <- 0; c<4; c++)
  {
    imprima "digite elemento de A[" ,L+1, "][",  c+1,  "]: ";
    leia A[L][c];
    B[L][c] <- M[L][c] * 3;
  }
 }
 imprima "\n\t\tMATRIZ TRIPLO\n";
 para(L <- 0; L<3; L++)
 {
  para(c <- 0; c<4; c++)
  {  imprima B[L][c], "\t";    }
  imprima "\n";
 }
 imprima "\n";
fimprog
```

algoritmo 418

Entrar com valores inteiros para uma matriz A[4] [4] e para uma matriz B[4] [4]. Gerar e imprimir a matriz SOMA[4] [4].

```
prog matriz17
 int A[4][4],B[4][4], SOMA[4][4], L,c;
 para(L <- 0; L<4; L++)
 {
 para(c <- 0; c<4; c++)
 {
   imprima "digite elemento de A[", L +1," ][", c+1, "] : ",;
   leia A[L][c];
  }
 }
 para(L <- 0; L<4; L++)
 {
 para(c <- 0; c<4; c++)
 {
   imprima "digite elemento de B[", L +1," ][", c+1, "] : ",;
   leia B[L][c];
   SOMA[L][c] <- A[L][c] + B[L][c];
  }
```

```
 }
 imprima "\n\tMATRIZ SOMA\n";
 para(L <- 0; L<4; L++)
 {
  para(c <- 0; c<4; c++)
  { imprima "\t", SOMA[L][c];}
  imprima "\n";
 }
 imprima "\n";
fimprog
```

algoritmo 419

Entrar com valores para duas matrizes inteiras de ordem cinco. Gerar e imprimir a matriz diferença.

```
prog matriz18
 int A[5][5],B[5][5], DIF[5][5], L,c;
 para(L <- 0; L<5; L++)
 {
   para(c <- 0; c<5; c++)
   {
     imprima "\ndigite elemento de A[", L+1, "][",c+1,"]: ";
     leia A[L][c];
   }
 }
 para(L <- 0; L<5; L++)
 {
  para(c <- 0; c<5;  c++)
  {
   imprima "\ndigite elemento de B[", L+1, "][",c+1,"]: ";
   leia B[L][c];
   DIF[L][c] <- A[L][c] - B[L][c];
   }
 }
 imprima "\n\tMATRIZ DIFERENCA\n";
 para(L <- 0; L<5; L++)
 {
  para(c <- 0; c<5; c++)
  {  imprima "\t", DIF[L][c]; }
  imprima "\n";
 }
 imprima "\n";
fimprog
```

algoritmo 420

Ler uma matriz 4 × 5 de inteiros, calcular e imprimir a soma de todos os seus elementos.

```
 }
 imprima "\n";
fimprog
```

algoritmo 424

Entrar com valores para uma matriz M_{2x2}. Calcular e imprimir o determinante.

Para cálculo do determinante de uma matriz de ordem 2, faça:

$$M = \begin{bmatrix} 6 & 7 \\ 2 & 4 \end{bmatrix}$$

multiplique os elementos da DP;
multiplique os elementos da DS e subtraia:
det = $6 \times 4 - 7 \times 2$

```
prog matriz23
 int L, c;
 real  det, M[2][2];
 para(L <- 0; L<2; L++)
 {
 para(c <- 0; c<2; c++)
 {
  imprima "digite elemento de M[" ,L+1, "][",  c+1,  "]: ";
  leia M[L][c];
  }
 }
 para(L <- 0; L<2; L++)
 {
  para(c <- 0; c<2; c++)
  {  imprima M[L][c], "\t";    }
  imprima "\n";
 }
 #Calcula o determinante
 det <- M[0][0] * M[1][1]  -  M[0][1] * M[1][0];
 imprima "\nDeterminante:", det;
 imprima "\n";
fimprog
```

algoritmo 425

Uma floricultura conhecedora de sua clientela gostaria de fazer um algoritmo que pudesse controlar sempre um estoque mínimo de determinadas plantas, pois todo dia, pela manhã, o dono faz novas aquisições. Criar um algoritmo que deixe cadastrar 50 tipos de plantas e nunca deixar o estoque ficar abaixo do ideal.

```
prog matriz24
 int L,c, qtde[50][3],d;
 string nome[50];
 para(L<-0;L<50;L++)
```

```
 { imprima "\n\ndigite nome da  ", L + 1, "  planta: "; leia nome[L];
   imprima "\ndigite quantidade em estoque: ";leia qtde[L][0];
   imprima "\ndigite qunatidade ideal: "; leia qtde[L][1];
   se(qtde[L][0]<qtde[L][1])
   { qtde[L][2]<-qtde[L][1] - qtde[L][0];}
   senao
   {qtde[L][2]<-0;}
 }
 imprima "\nTotal de compras\n";
 para(L<-0;L<5;L++)
 { imprima "\n", nome[L]," - ", qtde[L][2] ;}
 imprima"\n";
fimprog
```

algoritmo 426

A gerente do cabeleireiro Sempre Bela tem uma tabela em que registra os "pes", as "maos" e o serviço de podologia das cinco manicures. Sabendo-se que cada uma ganha 50% do que faturou ao mês, criar um algoritmo que possa calcular e imprimir quanto cada uma vai receber, uma vez que não têm carteiras assinadas; os valores, respectivamente, são: R$10,00; R$15,00 e R$ 30,00.

```
prog matriz25
 int L,c, dados[5][3];
 string nome[5];
 real valor[5];
 para(L<-0;L<5;L++)
 { imprima "\n\ndigite nome da  ", L + 1, " manicure: "; leia nome[L];
   imprima "\ndigite numero de maos feitas: "; leia dados[L][0];
   imprima "\ndigite numero de pes feitos: "; leia dados[L][1];
   imprima "\ndigite numero de servicos de podologia feitos: "; leia dados[L][2];
   valor[L]<-dados[L][0]*10.00 +dados[L][1]*15.00 +dados[L][2]*30.00;
 }
 imprima "\nSalao Sempre Bela - Valor a receber\n";
 para(L<-0;L<5;L++)
 { imprima "\n", nome[L], " valor a receber R$ ",valor[L] * 0.5 ;}
 imprima "\n";
fimprog
```

algoritmo 427

Criar um algoritmo que leia e armazene os elementos de uma matriz inteira M(10,10) e imprimi-la. Troque, a seguir:

- *a segunda linha pela oitava linha;*
- *a quarta coluna pela décima coluna;*
- *a diagonal principal pela diagonal secundária.*

Imprima a nova matriz.

```
prog matriz26
 int m[10][10], n[10][10],L,c,d;
 para( L<-0; L<10; L++)
 {  para( c<-0; c<10;c++)
    {   imprima "\ndigite elemento ", L+1," – ", c+1, ":  "; leia
     m[L][c]; n[L][c] <- m[L][c] ; }
  }
 para( c<-0; c<10;c++)
 { n[7][c] <- m[1][c];  n[1][c] <- m[7][c]; }
 para( L<-0; L<10; L++)
 {   n[L][3] <-m[L][9]; n[L][9] <- m[L][3];  }
 d <- 9;
 para( L<-0; L<10; L++)
 {   n[L][d] <- m[L][L] ;  n[L][L] <- m[L][d];  d--; }
 imprima "\nMatriz original\n";
 para( L<-0; L<10; L++)
 {  para( c<-0; c<10;c++)
    {   imprima m[L][c], "\t";  }
 }
 imprima "\nNova matriz\n";
 para( L<-0; L<10; L++)
 {  para( c<-0; c<10;c++)
    {   imprima n[L][c],"\t";  }
   }
 imprima "\n";
fimprog
```

algoritmo 428

A matriz dados contém na 1ª coluna a matrícula do aluno; na 2ª, o sexo (0 para feminino e 1 para masculino); na 3ª, o código do curso, e na 4ª, o CR.

Faça um algoritmo que armazene esses dados sabendo-se que:

- *o código do curso e uma parte da matrícula: aascccnnn (aa ano, s semestre, ccc código do curso e nnn matrícula no curso)*

Um grupo empresarial resolveu premiar a aluna com CR mais alto de um curso cujo código deverá ser digitado. Suponha 10 alunos e que o CR é um nº inteiro.

```
prog matriz27
 int dados[10][4],L,codigo,pos, aux;
 para(L<-0;L<10;L++)
 { imprima "\ndigite matricula no formato aascccnnn: "; leia dados[L][0];
   aux <- dados[L][0] div 1000000; aux <- dados[L][0] -aux*1000000;
   dados[L][2] <- aux div 1000;
   imprima "\ndigite sexo (0 fem 1 masc): "; leia dados[L][1];
   imprima "\ndigite CR do aluno: "; leia dados[L][3]; }
```

```
 imprima "\ndigite codigo do curso: "; leia codigo;
 pos<-0; / *posicao inexistente*/
 para(L<-0;L<10;L++)
 { se(dados[L][2]==codigo && dados[L][1]==0)
   { pos <- L;}
   senao
   { se(dados[L][3]>dados[pos][3])
     { pos <- L;}
 }}}
 imprima "\nAluna premiada matricula: ", dados[pos][0], " teve CR: ",
           dados[pos][3]; # continuacao
 imprima "\n";
fimprog
```

algoritmo 429

Criar um algoritmo que possa armazenar as alturas de dez atletas de cinco delegações que participarão dos jogos de verão. Imprimir a maior altura de cada delegação.

```
prog matriz28
 int L, c,d;
 real altura[5][10], aux;
 string deleg[5], resp;
 para(L<-0;L<5;L++)
 {  imprima "\ndigite nome do pais:  ",L+1, ":  ";
    leia deleg[L] ;
    para(c<-0;c<10;c++)
    {  imprima "\ndigite altura do atleta : ",c+1, ":  "; leia
    altura[L][c]; }
 }
 para(L<-0;L<5;L++)
 { imprima "\n\n", deleg[L] ,"\n";
   para(c<-0;c<10;c++)
   { imprima altura[L][c], "\t", }
   imprima "\n\npressione enter";
   leia resp;
 }
 para(d<-0; d<5;d++)
 { para(L<-0; L<9; L++)
   { para(c<-L+1; c<10;c++)
    { se(altura[d][L]<altura[d][c])
     { aux<-altura[d][L];
       altura[d][L]<-altura[d][c];
       altura[d][c]<-aux;
      }
    }
```

```
  }
 }
 imprima "\n Relacoes dos atletas mais altos por delegacao\n";
 para(L<-0; L<5; L++)
 { imprima "\n", deleg[L]," : ", altura[L][0];}
 imprima "\n";
fimprog
```

algoritmo 430

A Viação José Maria Rodrigues tem na Rodoviária de Rio Novo uma tabela contendo os horários de partidas dos ônibus para Juiz de Fora nos sete dias da semana.

Criar um algoritmo que possa armazenar esses horários e os horários do dia quando forem solicitados pelo funcionário, sabendo-se que, no máximo, são dez horários.

```
prog matriz29
 int L,c, d, dia;
 real  hora[7][10], horario;
 string ds[7][15];
 para(L<-0;L<7;L++)
 { para(c<-0;c<10;c++)
   {hora[L][c]<-0.;}
 }
 para(L<-0;L<7;L++)
 {  d<-0;
   imprima "\n\ndigite dia da semana: ";leia ds[L];
   imprima "\ndigite  ", d+1, " horario (1 - 24) ou -1 para terminar: ";
   leia horario;
     enquanto(d<10 && horario > 0)
     { hora[L][d]<-horario; d++;
       imprima "\ndigite  ", d+1, " horario (1 - 24) ou -1 para terminar: ";
       leia horario;
     }
 }
 para(L<-0;L<7;L++)
 { imprima "\n", ds[L], "\t";
   d<-0;
   enquanto(hora[L][d] >0 && d <10)
   { imprima hora[L][d],  "\t"; d++; }
    imprima "\n";
 }
 imprima "\ndigite 1 dom  2 seg 3 ter 4 quar 5 quin 6 sex 7 sabado 0 sair: ";
 leia dia;
 enquanto (dia  0)
 {
   enquanto(dia <1 || dia >7)
   { imprima "\nDia invalido.";
```

```
    imprima "\ndigite 1 dom  2 seg 3 ter 4 quar 5 quin 6 sex 7 sabado: ";
     leia dia;
   }
    dia--;
   imprima "\n\nViacao Jose maria Rodrigues";
    imprima "\n\nHorarios de  ", ds[dia]  ,"\t";
    d<-0;
    enquanto(hora[dia][d] >0 && d <10)
   { imprima hora[dia][d], "\t"; d++; }
    imprima "\n";
    imprima "\ndigite 1 dom  2 seg 3 ter 4 quar 5 quin 6 sex 7 sabado 0
    sair: ";
   leia dia;
 }
 imprima "\n";
fimprog
```

algoritmo 431

*Criar um algoritmo que entre com valores inteiros para uma matriz **m** 3 × 3 e imprima a matriz final, conforme mostrado a seguir:*

↳ *A matriz gira 90º.*

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad \begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$$

```
prog matriz30
 int L, c, m[3][3], final[3][3];
 para( L<- 0; L<=2; L++)
 {
   para( c<- 0; c<=2; c++)
   { imprima "\ndigite elemento: ",L + 1," - ",c + 1," : "; leia  m[L][c];  }
 }
 imprima "\nmatriz original\n";
 para( L<- 0; L<=2; L++)
 {
    para( c<- 0; c<=2; c++)
    { imprima m[L][c], "\t";   }
    imprima "\n";
 }
 imprima "\nmatriz gira 90 \n";
 para( c<- 0; c<= 2; c++)
 {
   para( L<- 2; L>=0; L--)
```

```
    { imprima m[L][c], "\t";   }
    imprima "\n";
  }
  imprima "\n";
fimprog
```

algoritmo 432

Criar um algoritmo que entre com valores inteiros para uma matriz **m** *3 × 3 e imprima a matriz final, conforme mostrado a seguir:*

A matriz gira 180º.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$$

```
prog matriz31
 int L, c, m[3][3], final[3][3];
 para( L<- 0; L<=2; L++)
 {
   para( c<- 0; c<=2; c++)
   { imprima "\ndigite elemento: ",L + 1," - ",c + 1," : "; leia  m[L][c];  }
 }
 imprima "\nmatriz original\n";
 para( L<- 0; L<=2; L++)
 {
    para( c<- 0; c<=2; c++)
    { imprima m[L][c], "\t";  }
    imprima "\n";
 }
 imprima "\nmatriz gira 180 \n";
 para( L<- 2; L>= 0; L--)
 {
   para( c<- 2; c>=0; c--)
   { imprima m[L][c], "\t";}
   imprima "\n";
 }
 imprima "\n";
fimprog
```

algoritmo 433

*Criar um algoritmo que entre com valores inteiros para uma matriz **m** 3 × 3 e imprima a matriz final, conforme mostrado a seguir:*

↳ *A matriz gira 270º.*

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \qquad \begin{bmatrix} 3 & 6 & 9 \\ 2 & 5 & 8 \\ 1 & 4 & 7 \end{bmatrix}$$

```
prog matriz32
 int L, c, m[3][3], final[3][3];
 para( L<- 0; L<=2; L++)
 {
   para( c<- 0; c<=2; c++)
   { imprima "\ndigite elemento: ",L + 1," - ",c + 1," : "; leia  m[L][c];  }
 }
 imprima "\nmatriz original\n";
 para( L<- 0; L<=2; L++)
 {
    para( c<- 0; c<=2; c++)
    { imprima m[L][c], "\t";  }
    imprima "\n";
 }
 imprima "\nmatriz gira 270 \n";
 para( c<- 2; c>= 0; c--)
 { para( L<- 0; L<=2; L++)
   { imprima m[L][c], "\t";  }
   imprima "\n";
 }
  imprima "\n";
fimprog
```

algoritmo 434

Criar um algoritmo que leia uma matriz 5 × 5 inteira e apresente uma determinada linha da matriz, solicitada via teclado.

```
prog matriz33
 int n[5][5],L, c;
 para(L <- 0; L < 5; L++)
 {
  para(c <- 0; c < 5; c++)
  {
```

```
      imprima "digite elemento de N[" ,L+1, "][",  c+1,  "]: ";
      leia n[L][c];
   }
   }
 #Entra com o número da linha
 imprima "\nDigite numero da linha(1-5) ou -999 para acabar: ";
 leia L ;
 enquanto (  L <> -999  )
 {
   enquanto (  L  < 1  || L  > 5  )
   { imprima "\nDigite número da linha entre 1 – 5: ";    leia  L ;  }
   L <- L-1;
   imprima "\n Linha: ",L + 1, "\n";
   para(c <- 0; c < 5; c++)
   { imprima n[L][c], "\t"; }
   imprima "\nDigite número da linha(1-5) ou -999 para acabar: ";
  leia L ;
 }
 imprima "\n";
fimprog
```

algoritmo 435

Criar um algoritmo que leia uma matriz 5 × 5 inteira e apresente uma determinada coluna da matriz, solicitada via teclado.

```
prog matriz34
 int n[5][5],L, c;
 para(L <- 0; L < 5; L++)
 {
  para(c <- 0; c < 5; c++)
  {
     imprima "digite elemento de N[" ,L+1, "][",  c+1,  "]: ";
     leia n[L][c];
  }
  }
 #Entra com o número da coluna
 imprima "\nDigite número da coluna(1-5) ou -999 para acabar: ";
 leia c ;
 enquanto (  c < > -999  )
 {
   enquanto (  c  < 1 || c  > 5  )
   {
    imprima "\nDigite número da coluna entre 1 – 5: ";
   leia  c ;
   }
  c <- c-1;
  imprima "\n Coluna: ",c + 1, "\n";
```

```
  para(L<- 0; L < 5; c++)
  {  imprima n[L][c], "\n"; }
 imprima "\nDigite número da coluna(1-5) ou -999 para acabar: ";
 leia c ;
 }
 imprima "\n";
fimprog
```

algoritmo 436

Criar um algoritmo que leia e armazene os elementos de uma matriz M10 × 10 inteira e imprima a soma de todos os elementos das colunas pares.

```
prog matriz35
 int M[10][10],L, c, soma;
 para(L <- 0; L < 10; L++)
 {
     para(c <- 0; c < 10; c++)
     { imprima "\ndigite elemento de M[" ,L+1, "][",  c+1,  "]: ";
       leia M[L][c]; }
 }
 imprima  "\n\nToda matriz\n";
 para(L <- 0; L < 10; L++)
 {
    para(c <- 0; c < 10; c++)
    {    imprima "\t" ,M[L][c];  }
   imprima  "\n";
 }
 soma <- 0;
 imprima  "\n\tColunas pares\n";
 para(L <- 0; L < 10; L ++)
 { para(c <- 1; c < 10; c <- c+2)
   {   imprima  "\t", M[L][c], "\t";  soma <- soma + M[L][c] ;}
    imprima  "\n";
 }
 imprima  "\nSoma dos elementos das colunas pares: ", soma;
 imprima  "\n";
fimprog
```

algoritmo 437

Criar um algoritmo que leia e armazene os elementos de uma matriz M inteira 10 × 10 e imprima todos os elementos que estão em linhas pares e colunas ímpares.

```
prog matriz36
 int M[10][10],L, c;
 para(L <- 0; L < 10; L++)
 {
     para(c <- 0; c < 10; c++)
```

```
    { imprima  "\ndigite elemento de M[", L+1, "][", c+1,  "]: ";
      leia M[L][c]; }
 }
 imprima  "\n\nToda matriz\n";
 para(L <- 0; L < 10; L++)
 {
  para(c <- 0; c < 10; c++)
  {    imprima M[L][c], "\t"  ;  }
 imprima  "\n";
 }
 imprima  "\n\tLinhas Pares - Colunas impares\n";
 para(L <- 1; L < 10; L <- L+2)
 {  imprima "\n";
   para(c <- 0; c < 10; c <- c+2)
   {    imprima  M[L][c],"\t\t" ; }
 }
 imprima  "\n";
fimprog
```

algoritmo 438

Criar um algoritmo que leia e armazene os elementos de uma matriz M inteira 50 × 50 e gere/imprima o vetor DIF, que é a diferença entre os vetores A e B assim gerados:

- *vetor A: contém todos os elementos abaixo da DS*
- *vetor B: contém todos os elementos acima da DP*

$$DIF = A - B$$

1. Os vetores Dif, A e B precisam ser dimensionados PRECISAMENTE, isto é: nem mais nem menos do que o necessário.
2. Nos trechos que geram os vetores A e B, *não é permitido* o uso da Estrutura do SE.
3. A seguir, cálculo para o número de elementos do vetor DIF:

50 × 50 = 2500 elementos
2500 – 50 (elementos da diagonal) = 2450
2450 elementos fora da diagonal
2450/ 2 = 1225

```
prog matriz37
 int L, c, x, M[50][50], A[1225], B[1225], DIF[1225];
 para( L<- 0 ; L<= 49; L++)
 {
```

```
   para( c<- 0; c  <= 49; c++)
   {
       imprima "\nelemento: ", L, " - ", c, ":  ";    leia M[L][c]; }
 }
x <- 0;
para(L<-1;L< 50;L++)
{
    para(c<-50-L;c< 50;c++)
    {  A[x ] <- M[L][c];     x++; }
}
x <- 0;
para(L<- 0;L < 49;L++)
{
   para(c <- L+1;c < 50;c++)
    {    B[x ] <- M[L][c];    x++; }
}
# Gera o vetor diferença
para(L<- 0;L < 1225;L++)
{ DIF[ L ] <- A[ L] - B[ L ]; }
imprima "\nVETOR DIFERENÇA\n";
 para(L<- 0;L <= 1224;L++)
 { imprima "\n", DIF[ L] ; }
imprima "\n";
fimprog
```

algoritmo 439

Criar um algoritmo que leia e armazene os elementos de uma matriz M inteira 60 × 60 e gere/imprima o vetor SOMA, que é a soma entre os vetores A e B assim gerados:

- *vetor A: contém todos os elementos acima da DS*
- *vetor B: contém todos os elementos abaixo da DP*
- *SOMA = A + B*

Aqui está o cálculo para o número de elementos do vetor SOMA:

60 × 60 = 3600 elementos *3600 – 60 (elementos da diagonal) = 3540* *3540 elementos fora da diagonal* *3540/ 2 = 1770*

```
prog matriz38
 int L, c, x, M[60][60], A[1770], B[1770], SOMA[1770];
 para( L<- 0 ; L< 60; L++)
```

```
{
   para( c<- 0; c  < 60; c++)
   {
       imprima "\nelemento: ", L, " - ", c, ":  "; leia M[L][c]; }
 }
x <- 0;
para(L<-0;L< 59;L++)
{
    para(c<-0;c< 59 - L ;c++)
    {  A[x ] <- M[L][c];  x++; }
}
x <- 0;
para(L<- 1;L < 60;L++)
{
   para(c <-0 ;c < L;c++)
    {     B[x ] <- M[L][c]; x++; }
}
# Gera o vetor soma
para(L<- 0;L < 1770;L++)
{ SOMA[ L ] <- A[ L] + B[ L ]; }
imprima "\nVETOR SOMA\n";
 para(L<- 0;L < 1770;L++)
 { imprima "\n", SOMA[ L] ; }
imprima "\n";
fimprog
```

algoritmo 440

Criar um algoritmo que carregue uma matriz 12 × 4 com os valores das vendas de uma loja, em que cada linha represente um mês do ano, e cada coluna, uma semana do mês. Calcule e imprima:

- *total vendido em cada mês do ano;*
- *total vendido em cada semana durante todo o ano;*
- *total vendido no ano.*

```
prog  matriz39
 real mat[12][4], totalmes, totalano, totalsem;
 int i, j;
 para( i <- 0; i <12; i++)
 {
   para( j<- 0; j <4; j++)
    { imprima "\nmes: ", i + 1, "  semana: ", j + 1, ":  ";
      leia mat[i][j]; }
 }
 totalano <- 0.;
 para( i <- 0; i <12; i++)
 { totalmes<- 0.;
```

```
   para( j <- 0; j <4; j++)
   {totalmes <- totalmes + mat[i][ j]; }
  imprima "\ntotal do mes  ", i + 1, "  = ", totalmes;
   totalano <- totalano + totalmes;
 }
 para( j <- 0; j <4; j++)
 {  totalsem <- 0.;
    para( i <- 0; i <12; i++)
    { totalsem <- totalsem + mat[i][j]; }
    imprima "\ntotal da semana  ", j + 1, "  = ", totalsem;
 }
 imprima  "\n\ntotal do ano = ",  totalano;
 imprima  "\n";
fimprog
```

algoritmo 441

Supondo que uma matriz apresente em cada linha o total de produtos vendidos ao mês por uma loja que trabalha com cinco tipos diferentes de produtos, construir um algoritmo que leia esse total e, ao final, apresente o total de produtos vendidos em cada mês e o total de vendas por ano por produto.

```
prog matriz40
 int L, c;
 real total[6][13];
 # Inicializa a 6ª  Linha
 para(c <- 0; c<13; c++)
 {  total[5][c ] <- 0.;}
 # Inicializa a coluna 12
 para(L <- 0; L<6; L++)
 {   total[L][12] <- 0.;}
 # Preenche a matriz total e calcula total linha
 para(L <- 0; L<5; L++)
 {
  para(c <- 0; c<12; c++)
  {
      imprima "\n valor do mes ", c,": ";
      leia total[L][c];
      # soma das colunas na 6ª linha
      total[5][c] <-total[5][c] + total[L][ c];
      # soma na 13ª coluna das linhas
      total[L][12] <- total[L][12] + total[L][ c];
   }
   total[5][12] <- total[5][12] + total[L][12] ;
 }
   # Lista totais por produto
 imprima "\n\nTotal de vendas ao ano por produto\n";
 para(L <- 0; L<5; L++)
```

```
{
   imprima "\nTotal de vendas do produto \n", L + 1;
  para(c <- 0; c<12; c++)
   {   imprima "\nmes: ",c + 1, ": ", total[L][c], ; }
   imprima "\notal Anual:", total[L][12];
}
# Lista totais por mes
para(c <- 0; c<12; c++)
 {  imprima "\nTOTAL DO MES ",c + 1, ": ",total[ 5][c];  }
imprima "\n Total Anual: ", total[5][12];
imprima "\n";
fimprog
```

algoritmo 442

A empresa Evite Desperdício tem registrados numa tabela (nome: consumo) os consumos mensais de energia elétrica dos anos 1990-1999. Cada linha representa um ANO e cada coluna um MÊS. Considerando esses dados, fazer um algoritmo que imprima:

a) o consumo médio em cada um dos meses destes dez anos.
b) o mês/ano em que a empresa gastou mais energia nestes dez anos.

```
prog matriz41
 int  consumo[10][12], L, c, mes, ano, maiorvalor;
 real  soma;
 para( L <- 0; L <= 9; L++)
 {
    para( c <- 0; c <= 11; c++)
    {
         imprima "\nentre com o consumo do ano:", 1990 + L , " mes: ", c
         + 1, ": ";
         leia consumo[L][c];
    }
 }
 para( c <- 0; c <= 11; c++)
 {
    soma <- 0.;
    para( L <- 0; L <= 9; L++)
    {   soma <- soma  +  consumo[L][c]; }
    imprima "\nconsumo médio do mes: ", c + 1, " no período 1990-1999: ",
    soma / 10;
 }
 maiorvalor <- 0;
 para( L <- 0; L <= 9; L++)
 {
     para( c <- 0; c <= 11; c++)
    {
        se( consumo[L][c] >  maiorvalor )
```

```
      {
         maiorvalor <- consumo[L][c];
         mes <- c;
         ano <- L;
       }
    }
 }
 imprima "\nO maior consumo foi de ",maiorvalor, "  no mes:", mes, "  e
 no ano:", ano:", ano  +  1990 ;
 imprima "\n";
fimprog
```

algoritmo 443

Um comerciante trabalha com 190 produtos. Sendo apaixonado pela Álgebra das Matrizes, resolveu criar um algoritmo com as seguintes matrizes:

- *uma matriz de nome código*$_{190 \times 2}$ *que armazena os códigos dos produtos e as posições dos valores de compra, na matriz vendas (sob a forma de um número de 2 casas; ex.: linha2, coluna3 → 23).*
  - *uma matriz de nome vendas*$_{??? \times ???}$*, com as seguintes características:*
  - *na DP, o percentual de vendas para os valores que se encontram na linha (o último elemento da DP deverá ser zero)*
  - *no triângulo superior da DP, os valores de compra*
  - *no triângulo inferior, diametralmente oposto em relação à DP, os valores de venda, calculados a partir do valor de compra e do percentual de venda, cujo valor se encontra, como já foi dito, no elemento da linha que está na DP.*

Ajude o comerciante a criar este algoritmo, sabendo-se que deverão ser digitados código do produto, valor de compra e percentual de venda, e possa imprimir uma listagem contendo código, valor de compra e valor de venda.

```
prog matriz42
 real vendas[20][20];
 int codigo[190][2], L,c,n, L1, c1;
 n <-0;
 para( L<- 0; L <  19;L++)
 { imprima "\n\ndigite o percentual de venda(0-100) para a linha:  " ,L+1;
   leia vendas[L][L];
   para( c <-L+1; c<20;c++)
   {
     imprima "\ndigite o valor de compra do produto : ", L+1,"- ", c+1,":
";
     leia vendas[L][c] ;
     imprima "\ndigite o codigo do produto:";
     leia codigo[n][0] ;
     codigo[n][1] <- L * 10 + c;
```

```
    vendas[c][L]  <- vendas[L][L]/100 * vendas[L][c] + vendas[L][c] ;
    n++;
  }
}
imprima "\n\ncodigo\tcompra\tvenda\n";
para( L<- 0; L <  n;L++)
{
  L1 <- codigo[L][1] div 10; /* recuperando a linha */
  c1 <- codigo[L][1] %10;    /* recuperando a coluna */
  imprima "\n", codigo[L][0],"\t",vendas[L1][c1],"\t", vendas[c1][L1] ;
}
imprima "\n";
fimprog
```

algoritmo 444

Criar um algoritmo que armazene valores inteiros para uma matriz de ordem quatro. Imprimir toda a matriz e o determinante da matriz.

O algoritmo deverá impedir a entrada do 0 (zero) para qualquer elemento que se encontre na DS ou abaixo da DS e só deverá permitir a entrada do 0 (zero) para todo elemento que se encontre acima da DS.

Considerações: A solução deste algoritmo se torna simples desde que você saiba que, quando uma matriz tem um triângulo de zeros, em relação à DS, o cálculo do determinante se faz da seguinte maneira:

- *multiplicam-se os elementos que estão na **DS***
- *se a matriz for de ordem par, será igual ao resultado do produto*
- *se a matriz for de ordem ímpar, será igual ao resultado do produto negativado*

```
prog matriz43
 int m[4][4], n[4][4], L, c,d;
 para( L <- 0; L < 4;  L++)
 {
   para( c <- 0; c < 4;  c++)
   { imprima "\ndigite elemento ",L + 1, "-", c+ 1, ": ";   leia m[L][c];
     se( L + c>= 3)
     {  enquanto( m[L][c]==0)
        { imprima "\nNAO PODE SER ZERO. Digite elemento ",L + 1, "-", c+
        1, ": ";
        leia m[L][c]; }
     }
     senao
     { enquanto( m[L][c] 0 )
       { imprima "\nSO PODE SER ZERO. Digite elemento ",L + 1, "-", c +
       1, ": ";
```

```
      leia m[L][c];    }
     }
   }
 }
 imprima "\nToda a matriz\n" ;
 para( L <- 0; L < 4;  L++)
 {
   para( c <- 0; c < 4;  c++)
    { imprima m[L][c],"\t" ; }
    imprima "\n";
 }
 d <- 1;
 para( L <- 0; L < 4;  L++)
 {     d <- d * m[L][3 -L];}
 imprima  "\n determinante:  ", d;
 imprima  "\n";
fimprog
```

algoritmo 445

Criar um algoritmo que declare uma matriz 5 × 5 e armazene em um vetor o maior elemento cadastrado em cada coluna da matriz e em um vetor b o menor elemento cadastrado em cada coluna da matriz. Imprima:

- *toda a matriz;*
- *o vetor a;*
- *o vetor b;*
- *(a + b)/2.*

```
prog matriz44
 int mat[5][5], a[5], b[5], maior, menor, cont, x, y;
 para( x <- 0; x < 5;  x++)
 {
    para( y <- 0; y < 5;  y++)
     { imprima "\ndigite elemento ",x + 1, "-", y+ 1, ": "; leia
     mat[x][y]; }
 }
 /* trecho de selecao dos menores e maiores elementos das colunas */
 para( y<- 0; y < 5;  y++)
 {    maior <- mat[0][y];
      menor <- mat[0][y];
    para( x <- 0; x < 5;  x++)
    {
      se(mat[x][y] > maior)
      {  maior <-  mat[x][y];}
      senao
        { se(mat[x][y] < menor)
```

```
            { menor <- mat[x][y];}
         }
       }
     a[y] <- maior;
     b[y] <-menor;
 }
 imprima "\nToda a matriz\n" ;
 para( x <- 0; x < 5;  x++)
 {
    para( y <- 0; y < 5;  y++)
     {
     imprima mat[x][y],"\t" ;
       }
     imprima "\n";
 }
 imprima "\nMaiores elementos das colunas\n";
 para( y<- 0; y < 5; y++)
 { imprima a[y], "\t";}
 imprima "\n\nMenores elementos das colunas";
 para( y<- 0; y < 5; y++)
 { imprima b[y], "\t";}
 imprima "\n\n Exibindo o resultado\n";
 para( y<- 0; y < 5; y++)
 { imprima (a[y] + b[y])/2., "\t";}
 imprima "\n";
fimprog
```

algoritmo 446

Entrar com valores para a matriz $A_{3 \times 4}$ e para matriz $B_{4 \times 5}$. Gerar e imprimir a matriz produto.

↳ *Para poder multiplicar duas matrizes, é necessário que o número de linhas da matriz multiplicando seja igual ao número de linhas da matriz multiplicadora.*

```
prog matriz45
 int A[3][4],B[4][5], PROD[3][5], L,c,m;
  para(L <- 0; L<3; L++)
 {
  para(c <- 0; c<4; c++)
  {
    imprima "\ndigite elemento de A[", L+1, "][",c+1,"]: ";
    leia A[L][c];
   }
 }
 para(L <- 0; L<5; L++)
```

```
{
  para(c <- 0; c<5; c++)
 { imprima "\ndigite elemento de B[", L+1, "]["",c+1,"]: ";
   leia B[L][c];
  }
}
para(L <- 0; L<3; L++)
{
  para(c <- 0; c<5;  c++)
  { PROD[L][c] <-0;
    para(m <- 0; m<4; m++)
    { PROD[L][c]<- PROD[L][c] + A[L][m] * B[m][c];    }
  }
}
imprima "\n\tMATRIZ PRODUTO\n";
para(L <- 0; L<3; L++)
{
   para(c <- 0; c<5; c++)
  { imprima PROD[L][c], "\t";  }
    imprima "\n";
}
imprima "\n";
fimprog
```

algoritmo 447

Uma certa fábrica produziu dois tipos de motores M1 e M2, nos meses de janeiro a dezembro; o número de motores foi registrado na Tabela 1.

O setor de controle de vendas tem uma tabela de custo e do lucro (em mil reais) obtidos com cada motor conforme Tabela 2.

Tabela 1

	M1	*M2*
jan	*30*	*20*
fev	*5*	*10*
.	.	.
dez	*18*	*25*

Tabela 2

	Custo	*Lucro*
M1	*10*	*3*
M2	*15*	*2*

Os números são exemplos.

Criar um algoritmo que leia a produção mensal, custo e lucro dos motores M1 e M2. Calcular e imprimir o custo e o lucro em cada um dos meses e o lucro anual.

DICA: Para saber o custo e o lucro nos meses de janeiro a dezembro, basta que se faça o produto matricial das duas tabelas.

```
prog matriz46
 real B[2][2], PROD[12][2], soma;
 int A[12][2],L,c,m;
 imprima "\nProducao dos motores\n";
  para(L <- 0; L<12; L++)
 {
    para(c <- 0; c<2; c++)
    { imprima "\ndigite numero de motores no mes ", L+1, " do tipo:
    ",c+1,": ";leia A[L][c]; }
 }
 para(L <- 0; L<2; L++)
 { imprima "\ndigite custo do motor ", L+1, ": "; leia B[L][0]; }
 para(L <- 0; L<2; L++)
 { imprima "\ndigite lucro do motor ", L+1, ": "; leia B[L][1]; }
 para(L <- 0; L<12; L++)
 {
   para(c <- 0; c<2;  c++)
   { PROD[L][c] <-0;
     para(m <- 0; m<2; m++)
     { PROD[L][c]<- PROD[L][c] + A[L][m] * B[m][c];   }
   }
 }
 imprima "\n\tMATRIZ CUSTO / LUCRO\n";
 para(L <- 0; L<12; L++)
 {
    para(c <- 0; c<2; c++)
   { imprima PROD[L][c], "\t";  }
     imprima "\n";
 }
 soma<- 0.;
 para(L <- 0; L<12; L++)
 {soma <- soma + PROD[L][1];}
 imprima "\n\nLucro anual: ", soma;
 imprima "\n";
fimprog
```

algoritmo 448

Em um concurso interno, para ocupar os cargos de diretor e gerente de uma empresa, concorreram dez pessoas. Essas pessoas fizeram três provas X, Y e Z. Como os cargos tinham perfis diferentes, os pesos das provas eram diferentes para cada cargo. Cada candidato recebeu um número de 0 até 9. Criar um algoritmo que possa imprimir os números dos candidatos que foram nomeados para diretor e para gerente. Se o primeiro lugar para o cargo de gerente for o mesmo do cargo de diretor, então, o segundo colocado será nomeado para gerente.

```
prog matriz47
 int L, C,m, maiorD, maiorG1, maiorG2,
pesos[3][2],provas[10][3],pontos[10][2] ;
```

```
para( L<- 0; L < 10 ; L++)
{
   para( C<- 0; C < 3 ; C++)
   {
  imprima "\naluno: ", L, " nota : ",C+1, ":  ";
  leia provas[L][C];
   }
}
para( L<- 0; L < 3 ; L++)
{
  para( C<- 0; C < 2 ; C++)
  {
 imprima "peso da prova :  ", L + 1, "  cargo: ", C+1;
 leia pesos[L][C];
  }
}
para( L<- 0; L < 10 ; L++)
{
  para( C<- 0; C < 3 ; C++)
  {
      imprima "\t",provas[L][C];
  }
  imprima "\n";
}
imprima "\n\n";
para( L<- 0; L < 3 ; L++)
{
   para( C<- 0; C < 2 ; C++)
   {    imprima "\t",pesos[L][C]; }
  imprima "\n";
}
para(L <- 0; L<10; L++)
{
     para(C <- 0; C<3; C++)
    {
 pontos[L][C] <-0;
 para(m <- 0; m<3; m++)
 { pontos[L][C] <- pontos[L][C]  + provas[L][m] * pesos[m][C]; }
    }
}
imprima "\n\tPONTOS DO CARGO DE DIRETOR\n";
para(L <- 0; L<10; L++)
{ imprima "\n", L + 1," - ", pontos[L][0] ; }
imprima "\n\tPONTOS DO CARGO DE GERENTE\n";
para(L <- 0; L<10; L++)
{ imprima "\n", L + 1," - ", pontos[L][1] ; }
maiorD<-0;
para(L <- 1; L<10; L++)
```

```
{  se(pontos[L][0]>pontos[maiorD][0])
   {maiorD<-L;}
}
imprima "\n DIRETOR - CANDIDADTO: ",maiorD+1;
se(pontos[0][1] > pontos[1][1])
{maiorG1<-0; maiorG2<-1;}
senao
{ maiorG1<-1; maiorG2<-0;}
para(L <- 2; L<10; L++)
{  se(pontos[L][1] > pontos[maiorG1][1])
  {maiorG2<-maiorG1; maiorG1<-L; }
  senao
  { se(pontos[L][1]><-pontos[maiorG2][1])
      {maiorG2<-L; }
  }
}
se(maiorD==maiorG1)
{ imprima "\nGERENTE - CANDIDADTO: ",maiorG2+1;}
senao
{ imprima "\nGERENTE - CANDIDADTO: ",maiorG1+1;}
imprima  "\n";
fimprog
```

algoritmo 449

Duzentos alunos fizeram duas provas. Cada prova foi corrigida por dois professores. Cada professor atribuiu os pesos que desejou às provas. Criar um algoritmo que imprima a soma dos pontos de todos os alunos, segundo o critério de cada professor. As notas serão sempre inteiras.

Entenda como deveriam ficar as matrizes desta questão:

	P1A	*P2A*	*P1B*	*P2B*
0				
1				
2				
3				
4				
5				
6				
7				
...				
199				

	Prof. A	*Prof. B*
peso P1		
peso P2		

	pontos Prof. A	*pontos Prof. B*
0		
1		
2		
3		
4		
5		
6		
7		
...		
199		

```
prog  matriz48
 int a[200][4],b[2][2], p[200][2],L, c,m;
 para(L<-0;L<200;L++)
  {
     para(c<-0; c<2;c++)
      {p[L][c]<-0;}
 }
 para(L<-0;L<200;L++)
 {
    para(c<-0;c<4;c++)
   { imprima "\ndigite nota:",L+1," -  ", c+1, ":  "; leia a[L][c];   }
 }
 para(L<-0;L<2;L++)
 {
   para(c<-0;c<2;c++)
   {  imprima "\ndigite peso do prova : ", L+1, "  do professor : ",c+1,
   ": "; leia b[L][c];  }
 }
 para(L<-0;L<200;L++)
 {
     c<-0;
    para(m<-0; m<2;m++)
    {p[L][c]<-p[L][c] + a[L][m] * b[m][c];}
      c<-1;
     para(m<-2; m<4;m++)
     {p[L][c]<-p[L][c] + a[L][m] * b[m-2][c];}
 }
 imprima "\nRelacao das notas finais segundo criterio dos professores\n";
 para(L<-0;L<200;L++)
 {
    para(c<-0;c<2;c++)
    { imprima p[L][c], "\t";   }
   imprima "\n";
 }
 imprima "\n";
fimprog
```

algoritmo 450

Uma editora envia remessas, para cada estado do Brasil e para o DF, sempre em pacotes com 100 livros. O número de pacotes enviados por vez depende de cada estado/DF. Observar as tabelas a seguir e fazer um algoritmo que deixe o usuário entrar com os dados para as duas matrizes e, depois, escolher se deseja imprimir todos os dados, os totais para um dos tipos, mensalmente, *totais de todos os tipos em um* mês. *Sabe-se que a segunda tabela contém o número de pacotes, estipulado pela editora, no envio de cada remessa para cada estado/DF por tipo; e a primeira contém o número de remessas feitas por mês para cada estado/DF, e que, em cada remessa, todos os tipos são enviados. A consulta poderá ser feita várias vezes.*

Tabela 1

	JAN	FEV	MAR
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
...			
26			
27			

Tabela 2

	1	2	3	4	5	6	7	8	9	10	...	26	27
tipo 1													
tipo 2													
tipo 3													
tipo 4													
tipo 5													
tipo 6													
tipo 7													

Tabela 3

1	ACRE
2	ALAGOAS
...	
26	TOCANTINS
27	DF

CONSIDERAÇÃO: A editora tem uma tabela onde cada estado é associado a um número de 1-26, em ordem alfabética, e o DF é o nº 27, conforme Tabela 3.

```
prog matriz49
 int  tabela2[7][27], tabela1[27][3], tabela[7][3],L, c, m, op, tipo, mes;
 string resp;
 para(L <-0; L < 7; L++)
 { para(c <- 0; c< 27 ; c++)
   { imprima "\ntipo:",  L+1, "  estado: ",  c+1, " :  "; leia tabela2[L]
   [c];}
 }
 para( L <-0; L < 27; L++)
 { para(c <- 0; c< 3 ; c++)
   {  imprima "\nestado: ",  L+1, "  mes : ",  c+1, " :  "; leia tabela1[L]
   [c]; }
 }
 para(L <- 0; L< 7; L++)
 { para(c <- 0 ; c < 3;c++)
   { tabela[L][c] <- 0 ;
     para(m <- 0; m < 27; m++)
     { tabela[L][c] <- tabela[L][c]+ tabela2[ L][m]* tabela1[ m][c];}
```

```
    }
  }
  faca
  {
    imprima "\n\n Escolha a impressao";
    imprima "\n 1 - Todos os dados";
    imprima "\n 2 - Um tipo mensalmente";
    imprima "\n 3 - Todos os tipos por mes";
    imprima "\n 4 - Sair";
    imprima "\n Opcao: ";
    leia op;
    escolha( op )
    { caso  1:
         imprima "\n\tJAN\tFEV\tMAR\n\n";
         para(L <- 0; L< 7; L++)
         { imprima L+1, "\t";
           para(c <- 0 ; c < 3;c++)
           { imprima  tabela[L][c], "\t",; }
           imprima "\n";
         }
         pare;
       caso 2:
         imprima "\n Digite tipo ( 1- 7): ";
         leia tipo ;
         enquanto( tipo < 1  ||  tipo > 7)
         {  imprima "\n; TIPO INVALIDO. Digite tipo (1 -7): ";  leia tipo ;}
         para(c <- 0 ; c < 3;c++)
         { imprima tabela[tipo][c], "\t",;    }
         imprima "\n";
         pare;
       caso 3:
          imprima "\n Digite mes ( 1 Jan 2 Fev 3 Mar ): ";
          leia mes ;
          enquanto( mes < 1 || mes > 3)
          { imprima "\n MES INVALIDO. Digite mes ( 1 Jan 2 Fev 3 Mar ):";
          leia mes ;  }
          para(L <- 0; L< 7; L++)
          { imprima "\n", tabela[L][mes] ; }
          pare;
        caso 4:
         imprima "\n FIM";
       pare;
       senao:
         imprima "\n Opcao Nao Disponivel";
      }
      imprima "\nPressione enter: "; leia resp;#  para dar uma pausa
  }
  enquanto(op  4)
  imprima "\n";
fimprog
```

algoritmo 451

Criar um algoritmo que deixe entrar com valores para uma matriz de ordem cinco e verificar se ela é ou não uma matriz triangular superior.

Matriz triangular superior é uma matriz onde todos os elementos de posições L < C são diferentes de 0 e todos os elementos de L > C são iguais a 0.

```
prog matriz50
 # solucao com triagulacao da matriz
 int m[5][5], L, c, ts;
 para(L <- 0 ; L < 5 ;L++)
 {
     para(c <- 0 ; c < 5 ;c++)
    {imprima "\nElemento: ", L+1, "-", c+1, ": "; leia m[L][c];  }
 }
 ts <- 1;
 L  <- 1;
 enquanto( L< 5  && ts ==1 )
 { c <- 0;
   enquanto(c < L && ts ==1 )
   {
     se( m[L][c] == 0 && ts ==1 )
      { ts <- 1; }
     senao
      { ts <- 0; }
     c++;
   }
   L++;
 }
 imprima "\n\nToda a Matriz\n";
 para(L <- 0 ; L < 5 ;L++)
 {
   para(c <- 0 ; c < 5 ;c++)
   { imprima m[L][c], "\t";
   imprima "\n";
 }
 se( ts ==1 )
 { imprima "\nTriangular Superior";}
 senao
 { imprima "\n Nao e Triangular Superior";}
 imprima "\n";
fimprog
```

algoritmo 452

Criar um algoritmo que deixe entrar com valores para uma matriz de ordem cinco e verifique se ela é ou não uma matriz triangular inferior.

Matriz triangular inferior é uma matriz onde todos os elementos de posições L > C são diferentes de 0 e todos os elementos de L < C são iguais a 0.

```
prog matriz51
 # solucao com triangulacao da matriz
 int m[5][5], L, c, ti;
 para(L <- 0 ; L < 5 ;L++)
 {
   para(c <- 0 ; c < 5 ;c++)
   {imprima "\nElemento: ", L+1, "-", c+1, ":  "; leia m[L][c]; }
 }
 ti <- 1;
 L <- 0;
 enquanto( L< 4  && ti ==1 )
 { c <- L + 1;
   enquanto(c < 5 && ti ==1 )
   {
     se( m[L][c] == 0 && ti ==1 )
       { ti <- 1; }
     senao
       { ti <- 0; }
     c++;
   }
   L++;
 }
 imprima "\n\nToda a Matriz\n";
 para(L <- 0 ; L < 5 ;L++)
 {
   para(c <- 0 ; c < 5 ;c++)
   { imprima m[L][c], "\t";
     imprima "\n";
 }
  se( ti ==1 )
 { imprima "\nTriangular Inferior ";}
  senao
 { imprima "\n Nao e Triangular Inferior ";}
 imprima "\n";
fimprog
```

algoritmo 453

Criar um algoritmo que deixe entrar com valores para uma matriz de ordem cinco e verifique se ela é ou não uma matriz identidade.

Matriz identidade é uma matriz onde todos os elementos de posições L > C e L < C são iguais a 0 e os elementos das posições L = C são iguais a 1.

```
prog matriz52
 int m[5][5], L, c, ti, ts;
 para(L <- 0 ; L < 5  ;L++)
 {
     para(c <- 0 ; c < 5  ;c++)
    {imprima "\nElemento: ", L+1, "-", c+1, ":  ";   leia m[L][c];  }
 }
 i <- 1;
 tsi <- 1;
 L <- 0;
  enquanto( L < 5 &&  tsi == 1 &&  i == 1 )
  {  c <- 0;
      enquanto( c < 5 &&  tsi == 1 &&  i == 1)
      {
        se( L == c )
         { se( m[L][c] == 1 && i == 1)
      { i <- 1; }
      senao
      { i <- 0;}
          }
         senao
         { se(m[L][c] == 0 &&  tsi == 1  )
      {  tsi <-1; }
      senao
      {  tsi <- 0;}
         }
         c++;
      }
      L++ ;
 }
 imprima "\n\nToda a Matriz\n";
 para(L <- 0 ; L < 5  ;L++)
 {
     para(c <- 0 ; c < 5  ;c++)
     {  imprima m[L][c], "\t";
      imprima "\n";
 }
```

```
 se( tsi== 1  &&  i== 1 )
 { imprima "\nMatriz Identidade";}
 senao
 { imprima "\nNao e Identidade ";}
 imprima "\n";
fimprog
```

algoritmo 454

As tabelas dadas a seguir contêm vários itens que estão estocados em vários armazéns de uma companhia. É fornecido também o custo de cada um dos produtos armazenados.

	PRODUTO 1	*PRODUTO 2*	*PRODUTO 3*
Armazém 1	*1200*	*3700*	*3737*
Armazém 2	*1400*	*4210*	*4224*
Armazém 3	*2000*	*2240*	*2444*

Custo	
PRODUTO 1	*260,00*
PRODUTO 2	*420,00*
PRODUTO 3	*330,00*

Criar um algoritmo que:

- *leia o estoque inicial;*
- *determine e imprima quantos itens estão armazenados em cada armazém;*
- *qual o armazém que possui a maior quantidade de produto 2;*
- *o custo total de:*

- *cada produto em cada armazém;*
- *custo total de cada armazém;*
- *cada produto em todos os armazéns.*

```
prog matriz53
 int prod[3][3], tota[3],L,c, pos;
 real custo[3],custop[3],ca[3],total, custogeral;
 string  resp;
 para(L<-0;L<3;L++)
 {ca[L]<-0.; tota[L]<-0; custop[L]<-0.;}
 custogeral<-0.;
 para(L<-0;L<3;L++)
 { imprima "\ndigite preco do produto: ", L+1, ": "; leia custo[L];}
```

```
 { se( flag  == 1)
   { imprima "\n Eleitores por faixa: 1 (menor que 18)   2 ( 18-70)   3
(maior que 70) :";
     para( L<- 0; L <= 2 ; L++)
     {   imprima "\n Total de pessoas na faixa  ",L + 1, " :  ", faixa[L]
 ;   }
   }
   senao
   { imprima "\nDados não cadastrados"; }
 }
 senao
 {se(op ==3)
  { se( flag  == 1)
   { menor<- 0;
     para( L<- 1; L <= 2 ; L++)
     {
         se(faixa[L] < faixa[menor])
      {  menor<- L;}
     }
     imprima "\nFaixa com menor número de eleitores: ", menor;
   }
   senao
   { imprima "\nDados não cadastrados"; }
 }
 senao
 { se(op ==4)
   {imprima "\nSaindo";}
   senao
   {imprima "\nOpcao Invalida";}
  }}}
 }
 enquanto( op < >  4)
 imprima "\n";
fimprog
```

algoritmo 456

Criar um algoritmo que funcione de acordo com o menu a seguir:

> *MENU MATRIZ*
> *1. Entrar com valores para uma matriz 10 X 10*
> *2. Ordenar por linha*
> *3. Imprimir a soma dos números pares abaixo da DS*
> *4. Sair*
> *OPÇÃO:*

```
prog  matriz55
int L, C, flag,op,Li,i,j,aux, mat[10][10], soma ;
flag <-0;
faca
{
  imprima "\n\n\n";
  imprima "\n MENU MATRIZ \n";
  imprima "\n1 - Entra com valores for uma matriz 10 X 10";
  imprima "\n2 - Ordena por linha ";
  imprima "\n3 - Imprime a soma dos numeros pares abaixo da DS ";
  imprima "\n4 - Sai do algoritmo";
  imprima "\nOPCAO:";
  leia op;
  se( op == 1)
  {flag<-1;
    para( L<- 0; L < 10 ; L++)
    {
      para( C<- 0; C < 10 ; C++)
      {
        imprima "\nelemento linha: ",  L + 1, " coluna:",C + 1," : ";
        leia mat[L][C];
      }
    }
  }
  senao
  { se(op == 2)
    { se(flag == 0)
     {imprima "\n NAO TEM DADOS CADASTRADOS";}
     senao
     {
       imprima "\ndigite linha (1-10): ";
       leia Li;
       enquanto( Li < 1 || Li > 10 )
       { imprima "\ndigite linha (1-10): ";
         leia Li;
       }
       Li--;
       para(  C <- 0;  C<=9; C++)
       { imprima "\t",mat[Li][C]; }
       imprima "\n";
       para( i<- 0; i < 9 ; i++)
       {
         para(j<- i + 1; j < 10; j++)
         {
           se(mat[Li][i] > mat[Li][j]  )
           {
             aux <- mat[Li][i];
             mat[Li][i]<- mat[Li][j];
              mat[Li][j] <- aux;
           }
```

```
          }
        }
        para(  C <- 0;  C<=9; C++)
        {  imprima "\t",mat[Li][C]; }
        imprima "\n";
      }
    }
    senao
    { se(op == 3)
     { se(flag == 0)
      { imprima "\n NAO TEM DADOS CADASTRADOS";}
      senao
      { soma <- 0;
       para(L <- 1; L<= 9; L++)
       {
        para(C<- 10-L; C<=9; C++)
        {
          se( mat[L][C]  %  2  == 0)
          { soma <- soma + mat[L][C]; }
         }
       }
       imprima "\nSOMA:  ", soma;
    }
 }
 senao
 { se( op ==4 )
    { imprima "\nSaindo do algoritmo"; }
    senao
    {imprima "\nOpcao nao disponivel"; }
  } } }
}
 enquanto( op <> 4 )
imprima "\n";
fimprog
```

algoritmo 457

Uma doceira faz dez tipos de tortas por dia. Em todas as tortas ela usa farinha de trigo, açúcar e leite, cujas medidas são em xícaras. Ela gostaria de fazer um algoritmo que funcionasse de acordo com o menu a seguir:

> *TORTAS SONHOS*
> *1. Entrar com as quantidades dos três ingredientes para todas as tortas*
> *2. Entrar com as quantidades de tortas que serão feitas, por tipo, no dia atual*
> *3. Calcular e imprimir totais de xícaras de leite, açúcar e farinha para todas as tortas*
> *4. Sair do programa*
> *Opção:*

```
prog matriz56
 int L, C, m, flag,op, tortas[10];
 real material[3][10] , materialfinal[3];
 flag <-0;
 faca
 {
 imprima "\n\n\n";
 imprima "\n TORTAS SONHOS \n";
 imprima "\n1 - 1 – Entra com as quantidades dos tres ingredientes para
 todas as tortas ";
 imprima "\n2 - Entra com a quantidade de tortas de cada tipo feitas
 por dia ";
 imprima "\n3 - Calcula e imprime a quantidade usada de farinha de trigo,
 acucar e leite ";
 imprima "\n4 - Sai do algoritmo";
 imprima "\nOPCAO:";
 leia op;
 se( op == 1)
 {flag<-1;
 imprima "\ningrediente 1 - Farinha de trigo; ingrediente 2  acucar;
 ingrediente3 - leite\n";
  para( L<- 0; L < 3 ; L++)
  {
   para( C<- 0; C < 10 ; C++)
   {
     imprima "\ningrediente ", L + 1, " torta: ", C  + 1;
     leia material[L][C] ;
   }
  }
 }
 senao
  { se(op == 2)
    { se(flag == 0)
      {imprima "\n NAO TEM DADOS CADASTRADOS";}
      senao
      {
       para( L<- 0; L < 10; L++)
       {
          imprima "\n quantidade de tortas número:  ", L + 1;
          leia tortas[L] ;
      }}}
      senao
      { se(op == 3)
        { se(flag = =0)
          {imprima "\n NAO TEM DADOS CADASTRADOS";}
          senao
          {
          para(L <- 0; L<3; L++)
          { materialfinal[L] <- 0.;
            para(m <- 0; m<10; m++)
```

```
        {
          materialfinal[L]<-materialfinal[L]+material[L][m]*tortas[m];
        }
   }
  imprima "\n\tMATERIAL NECESSÁRIO\n";
  imprima "\ningrediente 1: Farinha de trigo; ingrediente 2: acucar;
  ingrediente 3:  leite\n";
  para(L <- 0; L<3; L++)
  {
    imprima "\n", L + 1," - ", materialfinal[L] ; }
  }
 }
 senao
 {se(op==4)
  {saia;}
  senao
  {imprima "\nOPCAO NAO DISPONIVEL";}
 } } }
 }
 enquanto(op <> 4)
 imprima "\n";
fimprog
```

algoritmo 458

Uma loja tem dez vendedores. Deseja-se cadastrar o vendedor pelo nome. Cada vendedor terá um número que corresponde ao da posição no vetor: VENDEDOR. Os totais de vendas de cada vendedor serão registrados em uma matriz VENDAS (supondo 27 dias úteis). O algoritmo deverá funcionar através do menu a seguir:

LOJA VENDE BARATO

1. *Cadastrar nome vendedor*
2. *Listar todos os vendedores*
3. *Cadastrar venda diária*
4. *Calcular/listar total de um vendedor*
5. *Calcular/listar venda diária da loja*
6. *Calcular/listar total de vendas da loja*
7. *Listar nomes prêmio*
8. *Sair*

OPÇÃO

Considerações:

- *cadastra nome dos vendedores de uma vez;*
- *lista todos os vendedores de uma vez numerados;*
- *cadastra todos os totais dos vendedores de uma vez diariamente;*
- *entra-se com o número do vendedor e o dia-limite e é calculado o total de vendas até aquele dia do vendedor. É impresso o nome e o total de vendas;*

- *entra-se com o dia e é calculado o total de vendas da loja naquele dia, sendo impresso.*
- *calcula e lista o total ao final do mês.*
- *entra-se com o valor de venda para o prêmio do mês e lista todos os vendedores que tiveram venda igual ou superior ao valor.*

```
prog matriz57
 int opcao, dia, dialimite, numerovendedor, c, c1, d, flag, v[27];
 real premio, s, vendas[10][27];
 string vendedor[10][20], resp;
 para(c <-0; c<10; c++)
 { para(c1 <-0; c1<27; c1++)
   { vendas[c][c1]<-0.;}
 }
 para(c <-0; c<27; c++)
 { v[c]<- 0;}
 flag<-0;
 faca
 {
  imprima "\nLoja Vende Barato";
  imprima "\n1 - Cadastra nome-vendedor";
  imprima "\n2 - Lista todos os vendedores";
  imprima "\n3 - Cadastra venda diaria";
  imprima "\n4 - Calcula/Lista total de um vendedor";
  imprima "\n5 - Calcula/Lista venda diaria da loja";
  imprima "\n6 - Calcula/Lista total de vendas da loja";
  imprima "\n7 - Lista nomes-premio";
  imprima "\n8 - Sair";
  imprima "\nOpcao:";
  leia opcao;
  escolha (opcao)
  {
   caso 1:
    para(c <-0; c<10; c++)
    { imprima "\n vendedor: ", c+1, ":  "; leia vendedor[c]; }
    flag<-1;
    pare;
   caso 2:
    se(flag==0)
    {imprima "\nCadastre primeiro vendedor";}
    senao
    { imprima "\n\nRelacao dos vendedores\n";
      para(c <-0 ; c<10; c++)
      { imprima "\n", c+1," - ", vendedor[c];  }
    }
    pare;
   caso 3:
    se(flag==0)
```

```
        { imprima "\nCadastre primeiro vendedor";}
        senao
        { imprima "\nDigite dia (1 - 27):";
          leia dia;
          enquanto(dia <1 || dia >27)
          { imprima "\nDia invalido. Digite dia (1 - 27): ";
            leia dia;}
            dia--;
            se(v[dia] 0 )
            { imprima "\nDados ja cadastrados para este dia";}
            senao
            { para(c <-0 ; c<10; c++)
             { imprima "\ndigite vendas do vendedor:",c+1, ":  ";
                leia vendas[c][dia];  }
             v[dia]<-1;
            }
        }
      pare;
    caso 4:
      se(flag==0)
      { imprima "\nCadastre primeiro vendedor";}
      senao
       {
        imprima "\nDigite numero do vendedor(1 - 10): ";
        leia numerovendedor;
        enquanto(numerovendedor <1 || numerovendedor >10)
        { imprima "\nNumero invalido. Digite numero do vendedor(1 - 10): ";
          leia numerovendedor;
        }
        numerovendedor--;
        imprima "\nDigite dia limite(1 - 27): ";
        leia dialimite;
        enquanto(dialimite <1 || dialimite >27)
        { imprima "\nDia invalido. Digite dia limite(1 - 27): ";
          leia dialimite; }
        s<-0.;
        dialimite--;
        para(c <-0; c<= dialimite; c++)
        { s<-  s + vendas[numerovendedor][c ]; }
        imprima "\nTotal do vendedor: ", vendedor[numerovendedor], "=  "s;
      }
      pare;
    caso 5:
      se(flag==0)
      { imprima "\nCadastre primeiro vendedor";}
      senao
      { imprima "\nDigite dia (1 - 27):";
```

```
        leia dia;
        enquanto(dia <1 || dia >27)
        { imprima "\nDia invalido. Digite dia (1 - 27): ";
          leia dia;}
        dia--;
        se(v[dia]== 0 )
        { imprima "\nNada cadastrado para este dia";}
        senao
        { s<-0.;
          para(c <-0; c< 10; c++)
          { s<-  s + vendas[c][dia]; }
          imprima "\nTotal do dia  = ",  s;
        }
      }
      pare;
    caso 6:
      s<-0.;
      para(c <-0; c< 10; c++)
      { para(c1 <-0; c1< 27; c1++)
        { s<- s + vendas[c][c1]; }
      }
      imprima "\nTotal da loja  =  ",  s;
      pare;
    caso 7:
      imprima "\n Digite premio: ";
      leia premio;
      imprima "\nRelacao dos vendedores que ganharam o premio\n";
      para(c <-0; c< 10; c++)
      {
        s<-0.;
        para(c1 <-0; c1< 27; c1++)
        { s<-  s + vendas[c][c1];   }
        se( s >= premio )
        { imprima "\n", vendedor[c];}
      }
      pare;
    caso 8 :
      imprima "\n Saindo doprograma";
      pare;
     senao:
      imprima "\n Opcao nao disponivel") ;
   }
   imprima "\nPressione enter:"; leia resp;
  }
  enquanto( opcao  8)
  imprima "\n";
fimprog
```

Capítulo 6

Funções

Você teve oportunidade de trabalhar com várias funções predefinidas, isto é, funções que já estão prontas para serem usadas, como vimos anteriormente, em outros capítulos. Algumas são funções trigonométricas muito conhecidas da Matemática que já são consideradas como itens fundamentais da notação algorítmica, adotada neste texto, outras são strings mais ou menos comuns em várias linguagens.

Apesar de todas essas funções, muitas vezes perguntamos: por que não implementaram esta função ou aquela outra?

A resposta é simples: todas as linguagens permitem que você crie suas próprias funções. (O interpretador sugerido não permite e resolvemos seguir a mesma linha da linguagem C, como já fizemos no caso das matrizes e do comando escolha.)

CONCEITO

Função é um trecho de um algoritmo independente com um objetivo determinado, simplificando o entendimento do algoritmo e proporcionando ao algoritmo menores chances de erro e de complexidade.

VANTAGENS

Gostaríamos de começar com uma mensagem:

> Algoritmos são desenvolvidos por pessoas, verificados por outras pessoas e processados por uma máquina.

As funções, através da passagem de parâmetro e através do seu nome, permitem que sejam retornados valores à rotina chamadora (geralmente o algoritmo), e dessa forma esses valores poderão ser impressos ou atribuídos a uma variável ou podem servir em operações aritméticas entre outras.

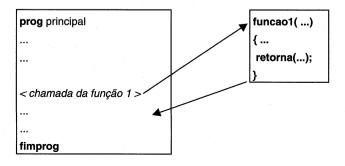
Os principais objetivos de uma função são:

- Dividir e estruturar um algoritmo em partes logicamente coerentes;
- Facilidade em testar os trechos em separado;
- O programador poderá criar sua própria biblioteca de funções, tornando sua programação mais eficiente uma vez que poderá fazer uso de funções por ele escritas em vários outros algoritmos com a vantagem de já terem sido testadas;
- Maior legibilidade de um algoritmo;
- Evitar que uma certa seqüência de comandos necessária em vários locais de um algoritmo tenha de ser escrita repetidamente nestes locais, diminuindo também o código-fonte.

Tudo isso justifica o uso de funções em nossos algoritmos, que no futuro se tornarão códigos em uma determinada linguagem.

CHAMADA DA FUNÇÃO

Não devemos ficar preocupados como acontecerá a chamada de funções, pois já fizemos uso de várias funções internas (funções do tradutor); chamaremos as funções feitas por nós da mesma maneira.

Quando uma função é chamada, o fluxo de controle é desviado para a função, no momento em que ela é ativada no algoritmo principal. Ao terminar a execução dos comandos da função, o fluxo de controle retorna ao comando seguinte àquele onde ela foi ativada, exatamente como na figura a seguir:

ESTRUTURA DE UMA FUNÇÃO

Uma função, como um algoritmo, é um bloco contendo início e fim, sendo identificada por um nome, pelo qual será referenciada em qualquer parte e em qualquer momento do algoritmo. A função serve para executar tarefas menores como ler, calcular, determinar o maior/menor valor entre uma lista de valores entre outras.

Após executar essas tarefas menores, a função retorna, ou não, um determinado valor para o algoritmo principal, no próprio nome da função, podendo ser numérico, lógico ou string.

Quando a função não retornar nada (nulo) usaremos o tipo void, pois é sugerido pelo comitê de padronização *ANSI*.

Dentro da função, podem ser declaradas variáveis que chamamos de variáveis locais, pois só são visíveis dentro da função.

Sintaxe da função:

```
<tipo de função> nome_da_função (parâmetros)
< declarações dos parâmetros >
{
 < declaração das variáveis locais
 comandos que formam o corpo da funcao
 retorna( valor ) ;
}
```

tipo de função	:	tipo de dado que a função dará retorno. Pode ser int, real, string ou void
nome da função	:	segue as mesmas regras de declaração de variáveis que já vimos
parâmetros	:	variáveis da função
declarações dos parâmetros	:	declarações de variáveis da função (tipo e nome). Podem vir juntos com os parâmetros
{	:	início da função
variáveis locais	:	declarações de variáveis que serão utilizadas dentro da função (tipo e nome)
corpo da função	:	seqüência de comandos
retorna(..)	:	o que vai ser retornado para o algoritmo ou não existe
}	:	fim da função

Considerações:
1. declarando uma função:

```
int fatorial(int  n)
{
ou
int fatorial( n)
int  n;
{
```

2. comando **retorna**
Este comando merece uma atenção especial, pois podemos fazê-lo de diferentes formas:

retorna(<expressão **>);** retorna o valor da expressão para o algoritmo
retorna; não retorna nada

3. A chamada da função dentro do algoritmo:

```
prog <nome_do_algoritmo>
<declaração das variáveis do algoritmo>
...
variavel <- nomedafunção(<parâmetro>) ;
               ou
 nomedafunção(<parâmetro>);
...
fimprog
```

LOCALIZAÇÃO DAS FUNÇÕES

Colocaremos as funções após o algoritmo principal, mas, em algumas linguagens, elas poderão ser colocadas logo após as declarações das variáveis.

```
prog principal
...
fimprog

funcao1(...)
{...
}

funcao2(...)
{...
}
funcao3(...)
{...
}
```

EXERCÍCIOS – LISTA 7
LISTA DE FUNÇÕES

Acreditamos, apesar de tudo o que já falamos sobre funções, que você deve estar se questionando: sempre fiz meus algoritmos e não usei funções e funcionou. Por que usá-las agora?

Essa é uma pergunta muito comum e só com exemplos poderemos mostrar as vantagens já faladas, mas, por ora, vale a pena lembrar: quantas vezes você já copiou um trecho de um algoritmo para ser usado em outro?

Como função é uma parte muito importante em algoritmos, procuramos não nos ater às possibilidades do interpretador sugerido, embora, sempre que foi possível, o façamos.

Vamos começar com um exemplo que irá convencê-lo de imediato.

algoritmo 459

Criar um algoritmo que produza a saída a seguir, usando uma função do tipo void:

algoritmo sem uso da função

```
prog  fun1
 int a;
 imprima "\n";
 para(a<<-1;a<=12;a++)
 {imprima "*"; }
 imprima "\nANITA LOPES";
 imprima "\n";
 para(a<-1;a<=12;a++)
 {imprima "*"; }
 imprima "\nGUTO GARCIA";
 imprima "\n";
 para(a<-1;a<=12;a++)
 {imprima "*"; }
 imprima "\n";
fimprog
```

algoritmo com uso da função

```
prog  fun1
 void linha( );
 linha( );#  este simbolo  é (
 e ) juntos
 imprima "\nANITA LOPES";
 linha( );
 imprima "\nGUTO GARCIA";
 linha( );
 imprima "\n";
fimprog
```

```
void linha( )
{ int a;
  imprima "\n";
  para(a<-1;a<=15;a++)
  {imprima "*"; }
}
```

Verifique a clareza do algoritmo principal quando usamos função; e se precisarmos a cada linha trocar o caractere? Veja como é simples:

VÍDEO
***************   ANITA LOPES   ===============   GUTO GARCIA   ###############

```
prog  fun1
 void linha( );
 linha("*");
 imprima "\nANITA LOPES";
 linha("=");
 imprima "\nGUTO GARCIA";
 linha("#");
 imprima "\n";
fimprog
```

```
void linha(c)
string c;
{ int a;
  imprima "\n";
  para(a<-1;a<=15;a++)
 {imprima c; }
}
```

*Para conseguirmos trocar o caractere que seria impresso, tivemos de passá-lo como parâmetro para a função **linha** e criar uma variável **c** para poder recebê-lo.*

Achamos que com esse simples exemplo já conseguimos deixá-lo com vontade de usar funções. Vamos continuar.

algoritmo 460

Criar um algoritmo que possa entrar com três números e, para cada um, imprimir o dobro. Usar uma função que retorne valor.

VÍDEO
digite numero: 12 dobro: 24 digite numero: 45 dobro: 90 digite numero: 78 dobro: 156

```
prog fun2
 int a,c; int dob( );
 para(c<-1; c<=3; c++)
 {
  imprima "\n\ndigite numero: "; leia a;
  imprima "dobro: ",dob(a);
 }
 imprima "\n";
fimprog
```

```
int dob(int x)
{
   retorne( x * 2);
}
```

Nesse exemplo, dentro da função não criamos nenhuma variável e retornamos o resultado, utilizando a variável da função. No próximo exemplo, mostraremos como uma função como essa poderia ser feita de maneiras diferentes e produzir o mesmo resultado na tela.

algoritmo 461

Criar um algoritmo que possa entrar com três números e, para cada um, imprimir o dobro. Usar duas funções.

VÍDEO
****

```
********************
digite numero: 12

dobro: 24
********************
digite numero: 36

dobro: 72
********************
digite numero: 67

dobro: 134
********************
```

```
Solucao 1
prog fun3
 int a,c;  void linha( ); int dob( );
 para(c<-1;c<=3;c++)
 {
  linha( );
 imprima "\ndigite numero: "; leia a;
 imprima "\ndobro: ",dob(a);
 }
 linha( );
 imprima "\n";
fimprog
```

```
void linha( )
{
int a;
 imprima "\n";
 para(a<-1;a<=20;a++)
 {imprima "*"; }
}
```

```
int dob(int x)
{
  x <- x *2;
}
```

O valor retornou na própria variável da função.

```
Solucao 2
prog fun3
 int a,c, res;  void linha( ); int dob( );
 para(c<-1;c<=3;c++)
 {
  linha( );
 imprima "\ndigite numero: "; leia a;
 res <- dob(a) ;
 imprima "\ndobro: ",res;
 }
 linha( );
 imprima "\n";
fimprog
```

```
void linha( )
{
 int a;
 imprima "\n";
 para(a<-1;a<=20;a++)
 {imprima "*"; }
}
```

```
int dob(x)
int  x;
{
 int y;
  y<- x *2;
  retorna(y);
}
```

O valor retornou na variável y criada na função e armazenado na variável ***res*** *do algoritmo principal.*

```
Solucao 3
prog fun3
 int a,c;  void linha( ); void dob( );
 para(c<-1;c<=3;c++)
 {
  linha( );
 imprima "\ndigite numero: "; leia a;
 imprima "\ndobro: ";
  dob(a);
 }
 linha( );
 imprima "\n";
fimprog
```

```
void linha( )
{ int a;
 imprima "\n";
 para(a<-1;a<=20;a++)
 {imprima "*"; }
}
```

```
void dob(int x)
{
   imprima "\n", x *2;
}
```

---

*O valor **não** retornou. O dobro foi impresso na própria função. A função do tipo **void** é assim: não retorna nada. Atenção quando for usá-la.*

---

algoritmo 462

Criar um algoritmo que receba notas de três provas e calcular a sua média, mas utilizando uma função.

Solução 1

VÍDEO
digite nota 1: 5 digite nota 2: 6 digite nota 3: 7 media aritmetica e 7.00 media armazenada em variavel = 0.00

```
prog fun4
 real n1,n2,n3, m, media( );
 imprima "\ndigite nota 1: ";leia n1;
 imprima "\ndigite nota 2: ";leia n2;
 imprima "\ndigite nota 3: ";leia n3;
 imprima "\nmedia aritmetica e ",media(n1,n2,n3);
 m<-media(n1,n2,n3);
 imprima "\nmedia armazenada em variavel = ",m;
 imprima "\n";
fimprog
```

```
real media(x,y,z)
real x,y,z;
{
  real ma;
  ma<-(x + y + z)/3;
}
```

Solução 2

VÍDEO
digite nota 1: 5 digite nota 2: 6 digite nota 3: 7 media aritmetica e 6.00 media armazenada em variavel = 6.00

```
prog fun4
 real n1,n2,n3, m, media( );
 imprima "\ndigite nota 1: ";leia n1;
 imprima "\ndigite nota 2: ";leia n2;
 imprima "\ndigite nota 3: ";leia n3;
 imprima "\nmedia aritmetica e ",media(n1,n2,n3);
 m<-media(n1,n2,n3);
 imprima "\nmedia armazenada em variavel = ",m;
 imprima "\n";
fimprog
```

```
real media(x,y,z)
real x,y,z;
{
  real ma;
  ma<-(x + y + z)/3;
  retorna(ma);
}
```

*Por que na primeira solução não saiu o valor da média armazenada na variável? Resposta: porque a variável **ma** era uma variável local e, como seu valor não foi retornado, só a função pode mostrar.*

algoritmo 463

Criar uma função que converta graus para radianos.

```
prog  fun5
 real ang,r, radiano( );
 imprima "\ndigite angulo em graus: ";
 leia ang:
 r <- radiano(ang);
 imprima "\ngraus: ", ang, "    radianos:  ",r;
 imprima "\n";
fimprog
```

```
 real radiano( rang)
 real rang;
 {
  rang<- rang * pi /180;
  retorna(rang);
 }
```

algoritmo 464

Criar uma função que converta radianos para graus.

```
prog  fun6
 real ang,g, graus( );
 imprima "\ndigite angulo em radianos: ";
 leia ang;
 g <- graus(ang);
 imprima "\nradianos: ",  ang, "  graus: ", g;
 imprima "\n";
fimprog
```

```
 real graus( gang)
 real gang;
 {
   gang<- gang * 180 / pi ;
   retorna(gang);
 }
```

algoritmo 465

Criar um algoritmo que imprima o maior número, usando uma função do tipo void.

```
prog fun7
 int num1,num2; void verifica( );
 imprima "\n\nDigite  numero 1: ";
 leia num1;
```

```
 imprima "\n\nDigite  numero 2: ";
 leia num2;
 verifica(num1,num2);
 imprima "\n";
fimprog
```

```
void verifica(n1, n2)
int n1, n2;
{
 se(n1 < n2)
 { imprima "\n", n2;}
 senao
 { imprima "\n", n1;}
}
```

algoritmo 466

Criar um algoritmo que receba um número que corresponda a um mês do 1º trimestre e escreva o mês correspondente; caso o usuário digite o número fora do intervalo deverá aparecer inválido, mas utilizando uma função do tipo void.

```
prog  fun8
 int mesc; void mes( );
 imprima "\ndigite um numero que corresponda a um mes do 1 trimestre:
 ";leia mesc;
 mes(mesc);
 imprima "\n";
fimprog
```

```
void mes(int x)
{
 se(x==1)
  { imprima "\njaneiro";}
  senao
   { se(x==2)
    {imprima "\nfevereiro";}
    senao
    { se(x==3)
       { imprima "\nmarco";}
       senao
       { imprima "\nNao e do 1 trimestre";}
    }
  }
}
```

Quando retiramos o conjunto de testes do algoritmo principal, ficou mais fácil seu entendimento, daí a importância do uso da função.

algoritmo 467

Criar um algoritmo para calcular o logaritmo de três números em qualquer base, usando uma função.

VÍDEO
digite logaritmando: 8 digite base: 2 logaritmo: 3.00000 logaritmo: 3.00000 digite logaritmando: 100 digite base: 10 logaritmo: 2.00000 logaritmo: 2.00000 digite logaritmando: 9 digite base: 3 logaritmo: 2.00000 logaritmo: 2.00000

```
prog fun9
 real z,L, b; int c; real loga( );
 para(c<-1;c<=3;c++)
 {
  imprima "\ndigite logaritmando maior do que 0: ";leia L;
   enquanto(L<=0)
   { imprima "\nInvalido.Digite logaritmando maior do que 0: ";   leia L;}
  imprima "\ndigite base maior que 0 e diferente de 1: ";leia b;
   enquanto(b<0 || b==1)
  { imprima "\nInvalido.digite base maior que 0 e diferente de 1: "; leia b;}
  imprima "\nlogaritmo e ",loga(L,b);
   z<-loga(L,b); imprima "\nz = ",z;
 }
 imprima "\n";
fimprog
```

```
real loga(base,expo)
real base, expo;
{
 real i;
 i<-log(base)/log(expo);
 retorna(i);
}
```

↳ *Você não precisa armazenar em uma variável para imprimir depois o resultado da função, só fizemos isso para reforçar que quando retornamos o valor através de uma variável local, podemos armazenar esse valor em uma variável no algoritmo principal.*

algoritmo 468

Criar uma função que possa calcular a raiz de três números positivos em qualquer índice. (Esta é uma função importante, pois a maioria das linguagens só oferece a função raiz quadrada.) A saída deverá ser assim:

VÍDEO

```
digite radicando: -64
Radicando invalido.Digite radicando: 64
digite indice: 1
Indice invalido.Digite indice: 2

Raiz:    8.000


digite radicando: 1000
digite indice: 3

Raiz:   10.000

digite radicando: 243
digite indice: 5

Raiz:    3.000
```

```
prog fun10
 { int r,i,c; real res; real raizn( );
 para(c<-1;c<=3;c++)
 {
  imprima "\n\ndigite radicando: ";leia r;
  enquanto(r < 0)
  { imprima "Radicando invalido.Digite radicando: ";leia r; }
  imprima "digite indice: ";leia i;
  enquanto(i <= 1)
  { imprima "Indice invalido.Digite indice: ";leia i;}
   res<- raizn(r,i);
  imprima "\nRaiz: ",res;
 }
 imprima "\n";
fimprog
```

```
real raizn( base, expo)
int base, expo;
{
   real y;
   y <- exp(1./expo * log(base));
   retorna(y);
}
```

Vamos explicar como chegamos à formula da função:

1. **log: = x** e se b^x **= A**, podemos substituir e obter a seguinte expressão:

$$b^{\log_e^A} = A$$

2. se considerarmos que b é a base neperiana, representada pela letra **e**, teremos: $b^{\log_e^A} = A$ mas e^x se transformará na função ***exp(x)*** e $\log_e^A$ função ***log(A)***, então nossa expressão será:

exp(log(A))

3. mas se assumirmos que A^n logo:

$$e^{\log_e^{A^n}} = A^n$$

4. pela propriedade dos logaritmos, $\log_e^{A^n} = n.\log_e^A$, ***e aí, teremos:***
   - $e^{n.\log_e^A} = A^n$
   - $e^{n.\log(A)} = A^n$

*exp(n * log(A)) – expressão final da potência*

Neste momento, você deve estar se perguntando: mas eu queria radiciação e me apresentam potência?

Não se preocupe, pois lembre-se de que toda raiz sai por uma potência com expoente fracionário, logo:

$$\sqrt[n]{A}$$

será equivalente a:

exp(1/n * log(A)) – expressão final da radiciação

Guarde essa fórmula, pois algumas linguagens que não têm operador de potenciação.

algoritmo 469

Criar um algoritmo que calcule o fatorial de um número, usando uma função que receba um valor e retorne o fatorial desse valor.

```
prog fun11
 int x, num;
 imprima "\ndigite numero positivo: ";
 leia num;
 enquanto(num<=0)
 { imprima "\nInvalido.Digite numero positivo: "; leia num;}
 x<-fatorial(num);
 imprima "\nfatorial de   ",num,"  =  ", fat;
 imprima "\n";
fimprog
```

```
int fatorial(n)
int n;
{
  int i, fat;
 fat<-1;
  para(i<-1;i<=n;i++)
  {fat<-fat*i;}
  retorna(fat);
}
```

algoritmo 470

Criar uma função que verifique se um número é primo.

```
prog fun12
 int num, pri; int primo( );
 imprima "\ndigite um numero >0: ";
 leia num;
 pri<-primo(num);
 se(pri ==0)
 {imprima "\n e primo",numi;}
 senao
 {imprima "\nnao e primo",num;}
 imprima "\n";
fimprog
```

```
int primo(n)
int n;
{
int c,p;
c<-0;
p<-2;
```

```
enquanto( c == 0 && p <= n div 2)
{ se(n % p == 0)
{ c<-1; }
  p++;
}
retorna(c);
}
```

algoritmo 471

Criar uma função que verifique quantas vezes um número é divisível por outro.

```
prog fun13
 int num,num1,n , divisor( );
 imprima "\ndigite dividendo: ";
 leia num;
 imprima "\ndigite divisor: ";
 leia num1;
 enquanto(num1 > num)
 {imprima "\nINVALIDO.digite numero menor do o dividendo: ";leia num1; }
 /* chama a funcao divisor */
 n<-divisor(num,num1);
 se(n==0)
 {imprima "\nNenhuma vez"; }
 senao
 {imprima "\nNumero de vezes: ",n; }
 imprima "\n";
fimprog
```

```
int divisor(x,y)
int x,y;
{
 int r, n1;
  n1<-0;
  r<- x % y;
  enquanto(r== 0)
  {   n1++;
      x<-x div y;
      r<- x % y;
  }
 retorna(n1);
}
```

algoritmo 472

Um número é dito regular se sua decomposição em fatores primos apresenta apenas potências de 2, 3 e 5. Faça uma função que verifique se um número é ou não regular.

```
prog fun14
 int num,num1, ch,d,c1, n,  primo( ), divisor( );
 d<-2;
 n<-0;
 imprima "\ndigite numero: ";
 leia num1;
 num<-num1;
 enquanto(d<=5 &&  n==0)
 {  /* chama a funcao divisor */
  n<-divisor(num,d);
 /*procura proximo numero primo */
  ch<-0;
  enquanto(ch==0)
  {
     d++;
    /* chama a funcao primo */
    c1<-primo(d);
    se(c1==0)
    { ch<-1;}
    senao
    { ch<-0;}
  }
 /*fim da procura do proximo numero primo */
 }
 se( n== 0)
 {imprima "\n", num1, "  e numero regular";}
 senao
 {imprima "\n", num1, "  nao e numero regular"; }
 imprima "\n";
fimprog
```

```
 int primo(n2)
 int n2;
 {
  int c,p;
   c<-0;
   p<-2;
  enquanto( c == 0 && p <= n2 div 2)
  { se(n2 % p == 0)
    { c<-1; }
    p++;
  }
  retorna(c);
 }
```

```
int divisor(x,y)
int x,y;
{
 int r, n1;
  n1<-1;
  r<- x % y;
  enquanto(r== 0)
  {   n1<-0;
      x<-x div y;
      r<- x % y;
  }
 retorna(n1);
}
```

algoritmo 473

Criar uma função que receba um caractere como parâmetro e retorne 1, caso seja uma consoante e 0 em caso contrário.

```
prog fun15
 string c; int maiuscula( ), x;
 imprima "\nDigite letra : "; leia c;
 x<-maiuscula(c);
 se(x== 1)
 { imprima "\ne uma consoante ";}
 senao
 { imprima "\nnao e uma consoante";}
 imprima "\n";
fimprog
```

```
int maiuscula(letra)
string letra;
{
  se(letra< > "A" && letra< > "a" && letra< > "E" && letra<<|>> "e"
&& letra< > "I" &&  letra< > "i" &&  letra< > "O" && letra< > "o" &&
letra< > "U" && letra< > "u"  )
 { retorna(1);}
 senao
 {retorna(0);}
}
```

Se a linguagem apresentar uma função que converta letra minúscula para maiúscula, este teste ficará mais simples.

algoritmo 474

Criar uma função que receba um caractere como parâmetro e retorne 1, caso seja uma vogal, minúscula ou maiúscula, e 0 em caso contrário.

```
prog fun16
 string c; int vogal( ), x;
 imprima "\nDigite letra: ";leia c;
 x<-vogal(c);
 se(x== 1)
 {imprima "\ne uma vogal ";}senao
 { imprima "\nnao e uma vogal";}
 imprima "\n":
fimprog
```

```
int vogal(letra)
string letra;
{
 se(letra== "a" || letra
=="e"|| letra == "i"|| letra
== "o"|| letra == "u"||
letra== "A" || letra =="E"||
letra == "I"|| letra == "O"||
letra == "U")
{ retorna(1);}
senao
{retorna(0);}
}
```

Se a linguagem apresentar uma função que converta letra minúscula para maiúscula, este teste ficará mais simples.

algoritmo 475

Criar uma função que receba um caractere como parâmetro e retorne 1, caso seja uma vogal minúscula, e 0, caso contrário.

```
prog fun17
 string c; int vogal( ), x;
 imprima "\nDigite letra minuscula: ";leia c;
 enquanto(c <"a" || c >"z")
 { imprima "\nInvalida.Digite letra minuscula: ";  leia c; }
 x<-vogal(c);
 se(x== 1)
 {imprima "\ne uma vogal minuscula ";}
 senao
 { imprima "\nnao e uma vogal";}
 imprima "\n":
fimprog
```

```
int vogal(letra)
string letra;
{
 se(letra== "a" || letra
=="e"|| letra == "i"|| letra
== "o"|| letra == "u")
  { retorna(1);}
  senao
  {retorna(0);}
}
```

algoritmo 476

Criar uma função que receba um caractere como parâmetro. A função deve retornar 1, se o caractere for uma letra maiúscula, e 0 caso contrário.

```
prog fun18
 string c; int maiuscula( ), x;
 imprima "\ndigite letra: "; leia c;
 x<-maiuscula(c);
 se(x== 1)
 {imprima "\ne uma letra maisucula";}
 senao
 { imprima "\nnao e uma letra maisucula";}
 imprima "\n";
fimprog
```

```
int maiuscula(letra)
string letra;
{
 se(letra>="A" && letra <="Z")
 { retorna(1);}
 senao
 {retorna(0);}
}
```

Lembre-se de que já falamos sobre o código ASCII; logo, quando questionamos um intervalo para verificar se é ou não uma letra maiúscula, é porque as letras maiúsculas têm código no intervalo fechado 65-90.

algoritmo 477

Criar uma função que receba um número inteiro como parâmetro e retorne 1, se sua raiz quadrada é exata, e 0 em caso contrário.

```
prog fun19
 int num,rz, raiz( );
```

algoritmo 479

Criar uma função que calcule o número de combinações de n elementos p a p. A fórmula da combinação é a seguinte:

$$C_p^n = \frac{n!}{p! * (n-p)!}$$

```
prog fun21
 int n,c, p, comb( ); int fatorial( );
 imprima "\ndigite numero de elementos da combinacao: ";
 leia n;
 enquanto(n<2)
 {imprima "\nInvalido.Digite numero de elementos da combinacao: "; leia
 n; }
 imprima "\ndigite numero de elementos que serao agrupados: ";
 leia p;
 enquanto(p> n)
 {imprima "\nInvalido.Digite numero de elementos que serao agrupados: ";
 leia p; }
 c<-comb(n,p);
 imprima "\nNumero de combinacoes = ",c;
 imprima "\n";
fimprog
```

```
int fatorial(f)
int f;
{
 int i, fat;
 fat<-1;
 para(i<-2;i<=f;i++)
 {fat<-fat*i;}
 retorna(fat);
}
```

```
int comb(n1,p1)
int n1,p1;
{
 int c1;
 c1 <- fatorial(n1) / (
fatorial(p1) *   fatorial(
n1-p1) );
retorna(c1);
}
```

algoritmo 480

*Um número é **capicua** quando lido da esquerda para a direita ou da direita para a esquerda. O ano 2002 é um ano **capicua**. Elabore uma função que verifique essa característica.*

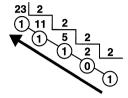
VÍDEO
digite numero: 123 123 - 321 Nao e um numero capicua digite numero: 2002 2002 - 2002 E un numero capicua

```
prog fun22
 int n, x, reverso( );
 imprima "\ndigite numero: ";
 leia n;
 x <- reverso(n);
 imprima "\n",n, " - ",x;
 se( x == n)
 { imprima "\nE un numero capicua";}
 senao
 { imprima "\nNao e  unm numero capicua";}
 imprima"\n";
fimprog
```

```
int reverso (int num)
{
int soma, r;
soma <- 0;
enquanto(num < > 0)
{
   r <- num % 10;
   soma <- soma * 10 + r;
   num <- num div 10;
}
retorna( soma);
}
```

algoritmo 481

Elaborar uma função que converta um número da base 10 para qualquer base entre 2 e 10, inclusive.

Exemplo na base 2:
Observe a figura ao lado que converte 23 (base 10) para 10111 (base 2).

VÍDEO
digite numero maior ou igual a 0: -12 Numero negativo.Digite numero maior ou igual a 0: 64 digite a base em que deseja representa-lo(2-10): 2 numero em decimal: 64 numero na base 2: 1000000 digite numero maior ou igual a 0: 64 digite a base em deseja representa-lo(2-10): 4 numero em decimal: 64 numero na base 4: 1000 digite numero maior ou igual a 0: 27 digite a base em deseja representa-lo(2-10): 5 numero em decimal: 27 numero na base 5: 102

```
prog fun23
 int num,base,c, converte( );
 imprima "\ndigite numero maior ou igual a 0: ";
 leia num;
 enquanto(num  < 0)
 { imprima "\nNumero negativo.Digite numero maior ou igual a 0: ";  leia
 num; }
 imprima "\ndigite a base em deseja representa-lo(2-10): ";leia base;
 enquanto(base <2 || base >10)
 { imprima "\nNao sei converter.Digite a base em deseja
 representa-lo(2-10): ";leia base;}
```

```
 c <- converte(num,base);
 imprima  "\nnumero em decimal:  ",num;
 imprima  "\nnumero na base "  base, " : ",c;
 imprima  "\n";
 imprima"\n";
fimprog
```

```
int converte(nnum,nbase)
int nnum,nbase;
{
 int nb, r,b;
  b<- 0;
  nb<- 0;
 enquanto( nnum >= nbase )
 {
    r <- nnum % nbase;
    nb <- nb + 10 ^b * r;
    nnum <- nnum div nbase;
    b++;
 }
  nb <- nb + 10 ^b * nnum;
retorna(nb);
}
```

algoritmo 482

Lembra-se do algoritmo que deixava entrar com três números e colocá-los em ordem crescente ou outros em que você precisou usar o trecho de troca? Vamos melhorá-lo. Que tal usarmos uma função para simplificar mais ainda?

```
prog  fun24
 int a,b,c,troca( );
 imprima "\nDigite  numero 1: ";
 leia a;
 imprima "\nDigite  numero 2: ";
 leia b;
 imprima "\nDigite  numero 3: ";
 leia c;
 se(a > b)
 { troca(a, b); }
 se(a>c)
 {troca(a, c);  }
 se( b >c )
 {troca(b, c);}
 imprima "\nOrdem Cresecnte: ", a, "  ",b, "  ",c ;
 imprima "\n";
fimprog
```

```
int troca(x, y)
int x, y;
{
  real aux;
  aux <- x;
  x <- y;
  y <- aux;
}
```

algoritmo 483

Criar uma função que receba como parâmetros dois vetores inteiros, e um inteiro indicando a quantidade de elementos que os dois vetores têm. A função deverá retornar o produto interno dos dois vetores.

Exemplo:
X <- (1,2,3,4)
Y <- (4,3,2,1)
X . Y <- 4 + 6 + 6 + 4 = 20

VÍDEO
VETOR A   VETOR B 2   1 3   2 4   3 5   4 produto interno: 40

```
prog fun25
 int L,c,aux , num[4],num1[4], produtointerno( );
 para(L <- 0;L<4;L++)
 { imprima "\n\nDigite 1 numero  ", L+1, ": "; leia num[L]; }
 para(L<- 0;L<4;L++)
 { imprima "\n\nDigite 2 numero ", L+1, ": "; leia num1[L]; }
 c <- produtointerno(num, num1,4);
 imprima "\nVETOR A\tVETORB\n";
 para(L <- 0;L<4;L++)
 { imprima "\n", num[L],"\t", num1[L]; }
 imprima "\n\nproduto interno: ",c;
 imprima "\n";
fimprog
```

```
int produtointerno(vet1, vet2,quant )
int vet1[ ], vet2[ ], quant;
{
  int prod, i;
  prod <- 0;
  para(i <-0;i<quant;i++)
  {   prod <-prod + (vet1[i] *
  vet2[i]); }
  retorna(prod);
}
```

algoritmo 484

Criar uma função que receba um vetor de inteiros e seu tamanho e retorne o menor dos elementos do vetor.

VÍDEO
VETOR 123 45 67 12 78 90 36 67 125 35 MENOR ELEMENTO: 12

```
prog fun26
 int L,c,num[10], menorelemento( );
 para(L<-0;L<10;L++)
 { imprima "\n\nDigite  numero  : ", L+1, ": "; leia num[L]; }
 c<-menorelemento(num,10);
  imprima "\nVETOR\n";
 para(L<-0;L<10;L++)
 { imprima "\n ",  num[L]; }
 imprima "\n\nMENOR ELEMENTO:  ", c;
 imprima "\n";
fimprog
```

```
int menorelemento (vet, tam)
int vet[ ], tam;
{
int  i, menor;
i <- 0;
menor <- vet[0];
para(i <-1; i<tam; i++)
{
  se( vet [i] < menor)
  {  menor <- vet[i];}
}
retorna (menor);
}
```

algoritmo 485

Criar uma função que receba um vetor de inteiros positivos, um vetor de caracteres e o tamanho (único) dos vetores. A função deve imprimir cada um dos caracteres do 2º vetor n vezes, onde n é o conteúdo da posição correspondente no vetor de inteiros.

VÍDEO
Digite numero 1: 6 Digite numero 2: 5 Digite numero 3: 4 Digite numero 4: 3 Digite numero 5: 2 Digite numero 6: 1 Digite caractere 1: e Digite caractere 2: s Digite caractere 3: t Digite caractere 4: u Digite caractere 5: d Digite caractere 6: e eeeeee sssss tttt uuu dd e

```
prog fun27
 int L,c,num[6]; string num1[6]; void multivetor( );
 para(L <- 0;L<6;L++)
 { imprima "Digite  numero  : ", L+1, ": "; leia num[L];}
 para(L <- 0;L<6;L++)
```

```
{  imprima "Digite caractere  : ", L+1, ": "; leia num1[L]; }
 imprima "\n";
 multivetor( num, num1,6);
 imprima "\n";
fimprog
```

```
void multivetor(vet1, vet2, tam )
int vet1[ ], tam; string vet2[ ];
{
 int i, w;
 para(i <- 0;i<tam;i++)
 {  imprima "\n";
    para(w <- 0;w < vet1[i]; w++)
    { imprima vet2[i]; }
 }
}
```

algoritmo 486

Criar uma função que receba o vetor e um elemento e verifique se o elemento está no vetor. O vetor está desordenado e tem dez elementos.

VÍDEO	
Digite numero 1: 23	*Digite numero 1: 23*
Digite numero 2: 45	*Digite numero 2: 4.5*
Digite numero 3: 6.7	*Digite numero 3: 6*
Digite numero 4: 8.9	*Digite numero 4: 78*
Digite numero 5: 90	*Digite numero 5: 90*
Digite numero 6: 12	*Digite numero 6: 13*
Digite numero 7: 56.7	*Digite numero 7: 202*
Digite numero 8: 4	*Digite numero 8: 15*
Digite numero 9: 16	*Digite numero 9: 6.7*
Digite numero 10: 12.5	*Digite numero 10: 5.5*
Digite numero de busca: 13	*Digite numero de busca: 13*
VETOR	*VETOR*
1 - 23.00	*1 - 23.00*
2 - 45.00	*2 - 4.50*
3 - 6.70	*3 - 6.00*
4 - 8.90	*4 - 78.00*
5 - 90.00	*5 - 90.00*
6 - 12.00	*6 - 13.00*
7 - 56.70	*7 - 202.00*
8 - 4.00	*8 - 15.00*
9 - 16.00	*9 - 6.70*
10 - 12.50	*10 - 5.50*
NAO ENCONTRADO	*ENCONTRADO*

```
prog   fun28
 int L,c,pertence( ); real n,num[10];
 para(L<-0;L<10;L++)
 { imprima "Digite  numero  : ", L+1, ": "; leia num[L];}
 imprima "Digite  numero de busca: ";
 leia n;
 c<-pertence(num,10,n);
 imprima "\nVETOR\n";
 para(L<-0;L<10;L++)
 { imprima "\n", L+1, " - ",  num[L]; }
 se(c==1)
 { imprima "\nENCONTRADO";}
 senao
 { imprima "\nNAO ENCONTRADO" ;}
 imprima "\n";
fimprog
```

```
int pertence( vetor,t,chave)
real vetor [ ],chave; int t;
{
  int achou, i;
  achou <- 0;
  i <- 0;
  enquanto(achou==0 && i <t )
  { se(vetor[i] == chave )
   {achou <- 1; }
   senao
   {i++; }
 }
 retorna( achou) ;
}
```

algoritmo 487

Criar uma função chamada inverte que receba um vetor de números inteiros como parâmetro e seu tamanho. A função deve inverter a ordem dos elementos do vetor de modo que o 1º vire o último e o 2º vire o penúltimo e assim sucessivamente.

VÍDEO
Digite numero 1: 23 Digite numero 2: 45 Digite numero 3: 67 Digite numero 4: 89 Digite numero 5: 12 Digite numero 6: 13

VÍDEO
Digite numero 7: 14 Digite numero 8: 15 Digite numero 9: 16 Digite numero 10: 17 VETOR 1 - 17 2 - 16 3 - 15 4 - 14 5 - 13 6 - 12 7 - 89 8 - 67 9 - 45 10 - 23

```
prog  fun29
 int L,inverte( ), n,num[10];
 para(L<-0;L<10;L++)
 { imprima "Digite  numero  : ", L+1, ": "; leia num[L];}
 inverte(num,10);
 imprima "\nVETOR\n";
 para(L<-0;L<10;L++)
 { imprima "\n", L+1, " – ",  num[L]; }
 imprima "\n";
fimprog
```

```
int inverte(vet , max )
int vet[ ], max;
{
  int k,i,aux;
  k<-max;
  para(i<-0;i<Max div 2;i++)
  {  aux <-vet [i];
     k--;;
     vet[ i ] <- vet [k];
     vet[k] <- aux;
  }
}
```

algoritmo 488

*Criar uma função que receba uma mensagem, seu tamanho e um caractere e retire todas as ocorrências desse caractere na mensagem colocando * em seu lugar. A função deve retornar o total de caracteres retirados.*

VÍDEO
digite mensagem:PRECISO FAZER OS 500 ALGORITMOS E COMECAR A CRIAR OS MEUS. digite letra: A total de trocas: 5 PRECISO F*ZER OS 500 *LGORITMOS E COMEC*R * CRI*R OS MEUS.

```
prog fun30
 string nome[10], letra;
 int  c, restantes( );
 imprima "\n\ndigite mensagem: ";
 leia nome;
 imprima "\ndigite letra: ";
 leia letra;
 c<- restantes(nome,strtam(nome),letra);
 imprima "\ntotal de trocas: ",c;
 imprima "\n",nome;
 imprima "\n";
fimprog
```

```
int restantes( vet, tam, x)
string vet[ ]; int tam; string x;
{ int i, cont;
cont=0;
para(i=0; i< tam; i++)
{ se(vet[i]==x)
 {  vet[i] = "*";
    cont++;;
}
}
retorna(cont);
}
```

```
#  sintaxe mais parecida com a
do interpretador sugerido
int restantes( vet, tam, x)
```

```
string vet[ ]; int tam; string x;
{ int i, cont; string  le, vet1;
 vet1="";
  cont=0;
  para(i <- 0; i< tam; i++)
  {
    le<-strelem(vet,i);
    se(le ==x)
    {vet1<-strconcat(vet1,"*");
    cont++;}
    senao
    {vet1<-strconcat(vet1,le);}
  }
  vet <- vet1;
}
```

algoritmo 489

Criar uma função que receba uma mensagem, seu tamanho e dois caracteres e substitua todas as ocorrências do 1º caractere pelo 2º caractere.

VÍDEO
digite mensagem: SE NAO APRENDI TODOS OS EXERCICIOS DE UM CAPITULO, NAO DEVO PASSAR PARA O OUTRO digite 1 letra: E digite 2 letra: K total de trocas: 6 SK NAO APRKNDI TODOS OS KXKRCICIOS DK UM CAPITULO, NAO DKVO PASSAR PARA O OUTRO

```
prog fun31
 string nome[10], letra1, letra2;
 int  c, restantes( );
 imprima "\nndigite mensagem:\n";
 leia nome;
 imprima "\ndigite 1 letra: ";
 leia letra1;
 imprima "\ndigite 2 letra: ";
 leia letra2;
 c<-restantes(nome,strtam(nome),letra1, letra2);
 imprima "\ntotal de trocas: ",c;
```

```
 imprima "\n",nome;
 imprima"\n";
fimprog
```

```
int restantes( vet, tam, x1, x2)
string vet[ ], x1, x2; int tam;
{
  int i, cont;
  cont <- 0;
  para(i <- 0; i< tam; i++)
  {
    se(vet[i]==x1)
    { vet[i] =<- x2; cont++;}
  }
  retorna(cont);
}
```

algoritmo 490

*Criar uma função que receba um vetor de caracteres, seu tamanho e um caractere e retire todas as ocorrências desse caractere no vetor, colocando * em seu lugar. A função deve retornar o total de caracteres retirados do vetor.*

VÍDEO

```
digite palavra em letras minusculas 1: escola
digite palavra em letras minusculas 2: sacada
digite palavra em letras minusculas 3: algoritmos
digite palavra em letras minusculas 4: festa
digite palavra em letras minusculas 5: escada
digite palavra em letras minusculas 6: cantar
digite palavra em letras minusculas 7: tocar
digite palavra em letras minusculas 8: estudar
digite palavra em letras minusculas 9: pancada
digite palavra em letras minusculas 10: cansei

1 - es*ola
2 - sa*ada
3 - algoritmos
4 - festa
5 - es*ada
6 - *antar
7 - to*ar
8 - estudar
9 - pan*ada
10 - *ansei
```

```
prog fun32
 string nome[10];
 int L, restantes( );
 para(L <- 0;L<10;L++)
 {
 imprima "digite palavra em letras minusculas ", L+1, ": ";
 leia nome[L];
 restantes(nome[L],strtam(nome[L] ),"c");
 }
 para(L <- 0;L<10;L++)
 { imprima "\n", L + 1, "  - "nome[L];}
 imprima"\n";
fimprog
```

```
int restantes( vet, tam, x)
string vet[ ]; int tam; string
x;
{
 int i, cont;
 cont <-0;
 para(i<-0; i< tam; i++)
 {
   se(vet[i]==x)
  {  vet[i] = "*" ;  cont++; }
}
  retorna(cont);
}
```

algoritmo 491

Quando precisamos criar uma tela com informações tabuladas com as strings, sempre temos problemas, pois os nomes têm tamanhos diferentes, assim como profissões e endereços. Fazer uma função que controle o número de caracteres na entrada de dados e acrescente espaços até completar um número predeterminado por você. Esse algoritmo trabalha com vetores.

VÍDEO

```
Digite  nome 1: ANITA LUIZA MACIEL LOPES

Nomes com ate 20 caracteres:

Digite  novamente: ANITA LOPES

Digite  prof: PROFESSORA UNIVERSITARIA
```

VÍDEO

```
Nomes com ate 15 caracteres:

Digite  novamente: PROFESSORA

Digite  endereco: RUA DO BISPO 83, RIO COMPRIDO BLOCO J

Nomes com ate 30 caracteres:

Digite  novamente: RUA DO BISPO 83

NOME                    PROFISSAO       ENDERECO

ANITA LOPES             PROFESSORA      RUA DO BISPO 83
GUTO GARCIA             PROFESSOR       CAMPUS NOVA AMERICA
JOAO BOND               DIPLOMATA       SUICA
```

```
prog fun33
 int L,c ,t;
 string nome[3], profe[3], ender[3];
 para(L<-0;L<3;L++)
 { imprima "\nDigite  nome  ", L+1, ": ";
     leia nome[L];  branco(nome[L],20);
     imprima "\nDigite  prof: ";
     leia profe[L];  branco(profe[L],15);
     imprima "\nDigite  endereco: ";
      leia ender[L] ;   branco(ender[L],30);
 }
 imprima "\nNOME\t\t\tPROFISSAO\tENDERECO\n";
 para(L <-0;L<3;L++)
 { imprima "\n", nome[L],"\t", profe[L],"\t",ender[L]; }
 imprima "\n";
fimprog
```

```
branco(n,tt1)
int tt1;string n[50];
{ int c,t;
  enquanto( strtam(n) >tt1)
  { imprima "\nNomes com ate  ", tt1, " caracteres: ";
      imprima "\n\nDigite  novamente: ";leia n ;
  }
  se(strtam(n) < > tt1)
  {    t <- tt1 - strtam(n);
       para(c<-1; c <= t; c++)
       { n <- strconcat(n, "ƀ"); }
 }
}
```

algoritmo 492

Criar uma função que receba um vetor de inteiros e seu tamanho e retorne 1 (um) se o vetor estiver ordenado de forma decrescente ou 0 (zero) se não estiver.

VÍDEO	
1 : 10	1 : 67
2 : 9	2 : 54
3 : 8	3 : 32
4 : 7	4 : 12
5 : 6	5 : 56
6 : 5	6 : 43
7 : 4	7 : 21
8 : 3	8 : 8
9 : 2	9 : 7
10 : 1	10 : 6
1	0

```
prog fun34
 int L,c,aux , num[10], busca( );
 para(L<-0;L<10;L++)
 {imprima "\nDigite  numero ", L+1, ": ";  leia num[L]; }
 c <- busca(num,10); #  veja observação
 para(L <- 0;L<10;L++)
 { imprima "\n", L+1, " - ", num[L];}
 imprima "\n\n",c;
 imprima "\n";
fimprog
```

```
int busca(vet,tam)
int vet[ ], tam;
{
 int i;
 para(i <- 0;i<tam-1;i++)
 {
    se(vet [i] < vet [ i +1])
    {  retorna(0); }
  }
  retorna(1);
}
```

↳ *Para que você possa testar esta função, seria preciso que já existisse um vetor com dados e você desconhecesse como ele estava ordenado. Por isso, colocamos as duas possibilidades.*

algoritmo 493

Criar uma função que receba um vetor e verifique se ele está ordenado, de forma crescente ou decrescente, ou se não está ordenado.

VÍDEO	VÍDEO	VÍDEO
Digite numero 1: 1	*Digite numero 1: 98*	*Digite numero 1: 9*
Digite numero 2: 2	*Digite numero 2: 76*	*Digite numero 2: 8*
Digite numero 3: 3	*Digite numero 3: 54*	*Digite numero 3: 7*
Digite numero 4: 4	*Digite numero 4: 32*	*Digite numero 4: 6*
Digite numero 5: 5	*Digite numero 5: 21*	*Digite numero 5: 5*
Digite numero 6: 6	*Digite numero 6: 19*	*Digite numero 6: 4*
Digite numero 7: 7	*Digite numero 7: 18*	*Digite numero 7: 3*
Digite numero 8: 8	*Digite numero 8: 15*	*Digite numero 8: 2*
Digite numero 9: 9	*Digite numero 9: 9*	*Digite numero 9: 1*
Digite numero 10: 10	*Digite numero 10: 3*	*Digite numero 10: 67*
ORDENACAO CRESCENTE	*ORDENACAO DECRESCENTE*	*NAO ESTA ORDENADO*

```
prog  fun35
 int L,c,aux , num[10], busca( ), busca1( ),verificaordem( );
 para(L<-0;L<10;L++)
 { imprima "Digite  numero  : ", L+1, ": "; leia num[L];}
 c<-verificaordem(num,10);
 se(c==1)
 { imprima "\nORDENACAO CRESCENTE"; }
 senao
 { se(c==2)
   { imprima "\nORDENACAO DECRESCENTE"; }
  senao
  { imprima "\nNAO ESTA ORDENADO"; }
 }
 imprima "\n";
fimprog
```

```
int busca(vet,tam)
int vet[ ], tam;
{
 int i,x,L;
 para(i<-0;i<tam-1;i++)
 {
   se(vet [i] < vet [i+1] )
   { x<-0; retorna(x); }
 }
 x<-1;retorna(x);
}
```

```
int busca1(vet,tam)
int vet[ ], tam;
{
 int i,x,L;
 para(i<-0;i<tam-1;i++)
 {
   se(vet [i] > vet [i+1])
   { x<-0; retorna(x);}
 }
 x<-1;retorna(x);
}
```

```
int verificaordem(vetor,t)
int vetor[ ],t;
{
 int r,res,L;
 r <- busca(vetor,t);
 se(r==1)
 { res<-2; retorna(res);}
 senao
 {  r<-busca1(vetor,t);
    se( r==1)
    { res<-1; retorna(res);}
    senao
    { res<-0; retorna(res);}
  }
}
```

algoritmo 494

Lembra-se do algoritmo com vetores que precisava ser ordenado e tinha vários dados para serem trocados e tivemos de repetir os trechos de troca? Vamos melhorá-lo agora usando a função ***troca****.*

VÍDEO
Digite nome: ANITA LOPES Digite endereco: RUA H Digite profissao: PROFESSORA Digite nome: GUTO GARCIA Digite endereco: RUA S Digite profissao: PROFESSOR Digite nome: ANA LUCIA Digite endereco: RUA A Digite profissao: PSICOLOGA ANA LUCIA RUA A PSICOLOGA ANITA LOPES RUA H PROFESSORA GUTO GARCIA RUA S PROFESSOR

```
prog fun36
 int L,c ,t,t1;
 string nome[3], prof[3], ender[3] , troca( );
 para(L<-0;L<3;L++)
 {
  imprima "\nDigite  nome: ";
  leia nome[L] ;
  imprima "\nDigite  endereco: ";
  leia ender[L] ;
  imprima "\nDigite  profissao: ";
  leia prof[L] ;
 }
 para(L<-0;L<2;L++)
 {
  para(c<-L+1;c<3;c++)
   { se( nome[L] > nome[c] )
   { troca(nome[L],nome[c] );
        troca(ender[L],ender[c] );
        troca(prof[L],prof[c] );
```

```
      }
   }
 }
 para(L<-0;L<3;L++)
 { imprima "\n", nome[L], "\t", ender[L], "\t",prof[L];}
 imprima "\n";
fimprog
```

```
string troca( n1,n2)
string n1,n2;
{
 string aux;
  aux <- n1;
  n1 <- n2;
  n2 <- aux;
}
```

algoritmo 495

Criar uma função que receba um vetor de caracteres e ordenar.

VÍDEO
Digite nome: PEDRO LOPES Digite nome: JOAO BOND Digite nome: GUTO GARCIA Digite nome: ANITA LOPES Digite nome: MARIA CORREA NOMES ORDENADOS 1 - ANITA LOPES 2 - GUTO GARCIA 3 - JOAO BOND 4 - MARIA CORREA 5 - PEDRO LOPES

```
prog  fun37
 int L,c ,t,t1;
 string nome[5][20], ordena( );
 para(L=0;L<5;L++)
 {   imprima "Digite  nome: ";    leia nome[L] ;}
 ordena(nome,5);
 imprima "\n\nNOMES ORDENADOS\n";
 para(L=0;L<5;L++)
 { imprima "\n",L + 1,"  - ",nome[L]; }
 imprima "\n";
fimprog
```

```
string ordena(vet, tam)
string  vet[ ]; int tam;
# vet[ ] repesenta que a funcao esta recebendo um vetor que tera a
dimensão tam
{
 int L1,c1;
 string aux;
 para(L1=0;L1<tam-1;L1++)
 {
   para(c1=L1+1;c1<tam;c1++)
   {
     se(vet[L1]>vet[c1])
     { aux <- vet[L1]; vet[L1] <- vet[c1]; vet[c1] <- aux;}
   }
  }
}
```

1. A função ordena torna o algoritmo mais legível do que a função troca.
2. Quando chamamos uma função e desejamos passar todo o conteúdo do vetor para essa função, simplesmente passamos o nome do vetor. Na função, uma variável declarada como vetor (**vet[ ]**) sem tamanho especificado receberá todo o conteúdo do vetor. Você conhecerá as especificações necessárias em cada linguagem em que for codificar este algoritmo.

algoritmo 496

Criar uma função que receba um vetor e um elemento e verifique se o elemento está no vetor. O vetor está ordenado crescentemente.

VÍDEO	VÍDEO
Digite numero de busca: 13	*Digite numero de busca: 13*
VETOR	*VETOR*
1 - 2.30	*1 - 1.00*
2 - 3.00	*2 - 8.00*
3 - 8.00	*3 - 12.00*
4 - 12.30	*4 - 13.00*
5 - 15.00	*5 - 26.00*
6 - 15.60	*6 - 34.00*
7 - 27.00	*7 - 35.00*
8 - 34.00	*8 - 45.00*
9 - 67.00	*9 - 67.00*
10 - 78.00	*10 - 89.00*
NAO ENCONTRADO	*ENCONTRADO*

```
prog  fun38
 int L,c, busca( );   real num[10],n, ordena( );
 para(L<-0;L<10;L++)
 { imprima "Digite  numero: ", L+1, ": "; leia num[L];}
 imprima "Digite  numero de busca: "; leia n;
 ordena(num,10);
 c<-busca(num,10,n);
 imprima "\nVETOR\n";
 para(L<-0;L<10;L++)
 { imprima "\n", L+1, " - ",  num[L]; }
 se(c==1)
 { imprima "\n\nENCONTRADO";}
 senao
 { imprima "\n\nNAO ENCONTRADO" ;}
 imprima "\n";
fimprog
```

```
int busca( vetor ,t, chave)
real  vetor[ ],chave; int t;
{
 int achou, i;
 achou <- 0;
 i <- 0;
 enquanto(i < t-1 && chave >
 vetor [ i ] )
 {  i++;}
 se(vetor [ i ] == chave )
 { achou<- 1; }
 retorna( achou );
}
```

```
real ordena(vet, tam)
real vet[ ]; int tam;
{
 int L1,c1,aux;
 para(L1<-0;L1<tam-1;L1++)
 {  para(c1<-L1+1;c1<tam;c1++)
   { se(vet[L1]>vet[c1] )
   { aux<-vet[L1];
vet[L1]<-vet[c1];vet[c1]<-aux; }
   }
 }
}
```

algoritmo 497

Criar uma função que implemente uma busca binária. Esta função recebe um vetor de inteiros, seu tamanho e uma chave e retorna a posição da chave no vetor. Lembre-se de que o vetor deverá estar ordenado.

VÍDEO	VÍDEO
Digite numero 1: 23 *Digite numero 2: 45* *Digite numero 3: 11* *Digite numero 4: 16* *Digite numero 5: 78* *Digite numero 6: 98* *Digite numero 7: 44* *Digite numero 8: 62* *Digite numero 9: 27* *Digite numero 10: 90* *Digite numero de busca: 89* *VETOR* *1 - 11* *2 - 16* *3 - 23* *4 - 27* *5 - 44* *6 - 45* *7 - 62* *8 - 78* *9 - 90* *10 - 98* *NÃO ENCONTRADO*	*Digite numero 2: 45* *Digite numero 3: 11* *Digite numero 4: 15* *Digite numero 5: 89* *Digite numero 6: 456* *Digite numero 7: 123* *Digite numero 8: 58* *Digite numero 9: 74* *Digite numero 10: 39* *Digite numero de busca: 89* *VETOR* *1 - 11* *2 - 15* *3 - 23* *4 - 39* *5 - 45* *6 - 58* *7 - 74* *8 - 89* *9 - 123* *10 - 456* *posicao no vetor: 8*

```
prog fun39
 int L,c,num[10],n,aux, busca( ), ordena( );
 para(L<-0;L<10;L++)
 { imprima "Digite  numero  : ", L+1, ": "; leia num[L];}
 imprima "Digite  numero de busca: "; leia n;
 ordena( num,10);
 c<-busca( num,10,n);
 imprima "\nVETOR\n";
 para(L<-0;L<10;L++)
 { imprima "\n", L+1, " - ",  num[L]; }
 c++;
 se(c< >0)
 { imprima "\n\nposicao no vetor:  ", c;}
```

```
 senao
 { imprima "\n\não ENCONTRADO" ;}
 imprima "\n";
fimprog
```

```
int busca(vet,tam, chave)
int vet[ ], tam, chave;
{
 int i, ini, meio, fim, n;
 ini <- 0;
 fim <- tam-1;
 enquanto(ini <= fim)
 {
 meio <- (ini+fim)/2;
 se(chave == vet[meio])
 {  retorna (meio);}
 senao
 { se(chave < vet[meio])
   { fim <- meio-1;}
   senao
   {  ini <- meio+1;}
  }
 }
 meio<-(-1) ;
 retorna (meio);
}
```

```
int ordena(vet, tam)
int vet[ ], tam;
{
 int L1,c1,aux;
 para(L1<-0;L1<tam-1;L1++)
 { para(c1<-L1+1;c1<tam;c1++)
   { se(vet[L1] > vet[c1] )
      { aux<-vet[L1];vet[L1]
      <-vet[c1];vet[c1]<-aux;}
   }
  }
}
```

Este algoritmo tem na rotina de busca uma grande vantagem sobre o anterior, porque a técnica da busca binária reduz o número de comparações, uma vez que vai "dividindo" o vetor em partes.

algoritmo 498

Criar uma função que leia vários números inteiros e o ultimo será o zero (existem no máximo 1000 números diferentes). Imprima o número de maior concorrência .

VÍDEO
digite numero: 13 digite numero: 2 digite numero: 2 digite numero: 3 digite numero: 5 digite numero: 2 digite numero: 2 digite numero: 2 digite numero: 0 1 - 13 2 - 2 3 - 3 4 - 5 Numero de maior concorrencia: 2 quantidade: 5

```
prog  fun40
 int pertence( ), maiorelemento( ); void ultimo( );
 ultimo( );
 imprima "\n";
fimprog
```

```
int pertence( vetor,t,chave)
int vetor [ ],chave; int t;
{
  int achou, i;
  achou <- 0;
  i <- 0;
  enquanto(achou==0 && i <t )
 {
   se(vetor[i] == chave )
```

```
    {achou <- 1; }
    senao
    {i++; }
  }
  se(achou == 1)
  { achou<-i;}
  senao
  {achou<-(-1);}
  retorna( achou) ;
}
```

```
int maiorelemento(vet, tam)
int vet[ ], tam;
{
 int  i, maior;
 i <- 0;
 maior <- 0;
 para(i <- 1; i<tam; i++)
 {
   se( vet [i] > vet[maior] )
   {  maior <- i;}
 }
 retorna(maior);
}
```

```
void ultimo( )
{int vet[1000],num;
   int total[1000],i,quant,x,r;
   para( i<-0;i<1000;i++)
   { total[i] <-  0; }
   imprima "digite numero: "; leia num) ;
   quant <- 0;
   enquanto(num < > 0 && quant <1000)
   { r<-pertence(vet,quant,num);
    se( r < > -1 )
    { total[r]++; }
    senao
    {  vet[quant] <- num; quant++; }
    imprima "\ndigite numero: "; leia num ;
 }
 x <- maiorelemento(total,quant);
 para( i<-0;i<quant;i++)
 { imprima "\n", i + 1, " – ",vet[i]; }
 imprima "\n\nNumero de maior concorrencia: ",
 vet[x]," quantidade: " , total[x]+1 ;
}
```

algoritmo 499

Criar um algoritmo que funcione de acordo com o menu a seguir, sabendo-se que os vetores têm dimensão 5. Os itens 1, 2 e 3 são funções:

MENU VETOR – FUNCAO *1. Dados do VETOR* *2. Ordena VETOR* *3. Imprime VETOR* *4. Sai do programa* *OPÇÃO:*

↳ *Apresentamos todas as telas para que você saiba o que pretendemos, inclusive com proteções para os trechos 2 e 3 que dependem do trecho 1. Você pode imprimir o vetor ordenado ou não.*

VÍDEO	VÍDEO	VÍDEO
MENU VETOR – FUNCAO *1 Dados do VETOR* *2 Ordena VETOR* *3 Imprime VETOR* *4 Sai do programa* *OPCAO:5* *Opcao invalida*	*MENU VETOR – FUNCAO* *1 Dados do VETOR* *2 Ordena VETOR* *3 Imprime VETOR* *4 Sai do programa* *OPCAO:2* *Escolha primeiro opcao 1*	*MENU VETOR – FUNCAO* *1 Dados do VETOR* *2 Ordena VETOR* *3 Imprime VETOR* *4 Sai do programa* *OPCAO:3* *Escolha primeiro opcao 1*

VÍDEO	VÍDEO	VÍDEO
MENU VETOR – FUNCAO *1 Dados do VETOR* *2 Ordena VETOR* *3 Imprime VETOR* *4 Sai do programa* *OPCAO:1* *Entrada do VETOR* *Digite numero 1: 23* *Digite numero 2: 45*	*MENU VETOR – FUNCAO* *1 Dados do VETOR* *2 Ordena VETOR* *3 Imprime VETOR* *4 Sai do programa* *OPCAO:2* *MENU VETOR – FUNCAO* *1 Dados do VETOR*	*MENU VETOR – FUNCAO* *1 Dados do VETOR* *2 Ordena VETOR* *3 Imprime VETOR* *4 Sai do programa* *OPCAO:4* *Saindo do algoritmo*

VÍDEO	VÍDEO	VÍDEO
Digite numero 3: 12 *Digite numero 4: 78* *Digite numero 5: 89* *MENU VETOR - FUNCAO* *1 Dados do VETOR* *2 Ordena VETOR* *3 Imprime VETOR* *4 Sai do programa* *OPCAO:3* *VETOR* *23* *45* *12* *78* *89*	*2 Ordena VETOR* *3 Imprime VETOR* *4 Sai do programa* *OPCAO:3* *VETOR* *12* *23* *45* *78* *89*	

```
prog fun41
 int  num[5],L, flag,op,ordena( ), entrada( ),imprime( );
 flag <- 0;
 para(L<- 0;L<5;L++)
 { num[L] <- 0; }
 faca
 {
    imprima  "\n\n\n";
    imprima  "\n MENU VETOR - FUNCAO \n";
    imprima  "\n1 Dados do VETOR";
    imprima "\n2 Ordena VETOR ";
    imprima "\n3 Imprime VETOR ";
    imprima "\n4 Sai do programa ";
    imprima "\nOPCAO:";
    leia op;
    escolha(op)
    {
      caso 1: entrada(num,5);
              flag <-1;
              pare;

      caso 2: se(flag == 1)
              { ordena(num,5); }
              senao
```

```
                {imprima "\nEscolha primeiro opcao 1";}
                pare;

        caso 3: se(flag == 1 )
                { imprime(num,5); }
                senao
                {imprima "\nEscolha primeiro opcao 1";}
                pare;

        caso 4: imprima "\nSaindo do Algoritmo";
                pare;

        senao: imprima "\nOpcao invalida";
    }

  }
  enquanto(op < > 4)
  imprima "\n";
fimprog
```

```
int entrada(vet,t)
int vet[ ],t;
{
 int L;
 imprima "\nEntrada do VETOR
 \n";
 para(L <- 0;L<t;L++)
 { imprima "Digite  numero  :
 ", L+1, ": "; leia vet[L];}
 retorna(vet);
}

int imprime(vet,t)
int vet[ ],t;
{
 int L;
 imprima "\nVETOR\n";
 para(L <- 0;L<t;L++)
 { imprima "\n", L+1, " - ",
 vetL]; }
 retorna(vet);
}
```

```
int ordena(vet, tam)
int vet[ ], tam;
{
```

```
  int L1,c1,aux;
  para(L1 <- 0;L1<tam-1;L1++)
  {   para(c1 <- L1+1;c1<tam;
  c1++)
      { se(vet[L1]>vet[c1])
       { aux <- vet[L1];
       vet[L1] <- vet[c1];
       vet[c1] <- aux;}
      }
  }
  retorna(vet);
}
```

algoritmo 500

Criar um algoritmo que funcione de acordo com o menu a seguir, sabendo-se que os vetores têm dimensão 5. Os itens 1, 2, 3 e 4 são funções:

VETORES

1. Dados do VETOR A
2. Dados do VETOR B
3. Imprime VETORES
4. Soma VETORES
5. Subtrai VETORES
6. Sai do programa
OPÇÃO:

Apresentamos todas as telas para que você saiba o que pretendemos, inclusive com proteções para os trechos 3, 4 e 5 que dependem dos trechos 1 e 2.

VÍDEO	VÍDEO
VETORES *1 Dados do VETOR A* *2 Dados do VETOR B* *3 Imprime VETORES* *4 Soma VETORES* *5 Subtrai VETORES* *6 Sai do programa* *OPCAO:7*	*VETORES* *1 Dados do VETOR A* *2 Dados do VETOR B* *3 Imprime VETORES* *4 Soma VETORES* *5 Subtrai VETORES* *6 Sai do programa* *OPCAO:4*

VÍDEO	VÍDEO
Opcao invalida *VETORES* *1 Dados do VETOR A* *2 Dados do VETOR B* *3 Imprime VETORES* *4 Soma VETORES* *5 Subtrai VETORES* *6 Sai do programa* *OPCAO:3* *Escolha primeiro opcoes 1 e 2*	*Escolha primeiro opcoes 1 e 2* *VETORES* *1 Dados do VETOR A* *2 Dados do VETOR B* *3 Imprime VETORES* *4 Soma VETORES* *5 Subtrai VETORES* *6 Sai do programa* *OPCAO:5* *Escolha primeiro opcoes 1 e 2*

VÍDEO	VÍDEO	VÍDEO
VETORES *1 Dados do VETOR A* *2 Dados do VETOR B* *3 Imprime VETORES* *4 Soma VETORES* *5 Subtrai VETORES* *6 Sai do programa* *OPCAO:1* *Entrada do VETOR A* *Digite numero 1: 10* *Digite numero 2: 9* *Digite numero 3: 8* *Digite numero 4: 7* *Digite numero 5: 6*	*VETORES* *1 Dados do VETOR A* *2 Dados do VETOR B* *3 Imprime VETORES* *4 Soma VETORES* *5 Subtrai VETORES* *6 Sai do programa* *OPCAO:2* *Entrada do VETOR B* *Digite numero 1: 5* *Digite numero 2: 4* *Digite numero 3: 3* *Digite numero 4: 2* *Digite numero 5: 1*	*VETORES* *1 Dados do VETOR A* *2 Dados do VETOR B* *3 Imprime VETORES* *4 Soma VETORES* *5 Subtrai VETORES* *6 Sai do programa* *OPCAO:3* *VETOR A* *10* *9* *8* *7* *6* *VETOR B* *5* *4* *3* *2* *1*

VÍDEO	VÍDEO	VÍDEO
VETORES	*VETORES*	*VETORES*
1 Dados do VETOR A	*1 Dados do VETOR A*	*1 Dados do VETOR A*
2 Dados do VETOR B	*2 Dados do VETOR B*	*2 Dados do VETOR B*
3 Imprime VETORES	*3 Imprime VETORES*	*3 Imprime VETORES*
4 Soma VETORES	*4 Soma VETORES*	*4 Soma VETORES*
5 Subtrai VETORES	*5 Subtrai VETORES*	*5 Subtrai VETORES*
6 Sai do programa	*6 Sai do programa*	*6 Sai do programa*
OPCAO:4	*OPCAO:5*	*OPCAO:6*
SOMA	*DIFERENCA*	*Saindo do Algoritmo*
15	*5*	
13	*5*	
11	*5*	
9	*5*	
7	*5*	

```
prog fun42
 int  num[5],num1[5],L, flag,flag1,op, entrada( ),imprime( ); void
 soma( ), subtrai( ) ;
 flag <-0; flag1<-0;
 para(L<-0;L<5;L++)
 { num[L]<-0; num1[L]<-0;}
 faca
 {
    imprima  "\n\n\n";
    imprima  "\n VETORES \n";
    imprima  "\n1 Dados do VETOR A";
    imprima "\n2 Dados do VETOR B ";
    imprima "\n3 Imprime VETORES ";
    imprima "\n4 Soma VETORES ";
    imprima "\n5 Subtrai VETORES ";
    imprima "\n6 Sai do programa ";
    imprima "\nOPCAO:";
    leia op;
    escolha(op)
    {
    caso 1: entrada(num,5, "A");flag<-1;
            pare;
    caso 2: entrada(num1,5,"B");flag1<-1;
            pare;
    caso 3: se(flag < >0 && flag1 < > 0)
            { imprime(num,5,"A"); imprime(num1,5,"B"); }
```

```
            senao
            {imprima "\nEscolha primeiro opcoes 1 e 2";}
            pare;
    caso 4: se(flag < >0 && flag1 < > 0)
            { soma(num,num1,5);}
            senao
            {imprima "\nEscolha primeiro opcoes 1 e 2";}
            pare;
    caso 5: se(flag < >0 && flag1 < > 0)
            { subtrai(num,num1,5); }
            senao
            { imprima "\nEscolha primeiro opcoes 1 e 2";}
            pare;
    caso 6: imprima "\nSaindo do Algoritmo";
            pare;
    senao: imprima "\nOpcao invalida";
  }
 }
 enquanto(op < > 6)
 imprima "\n";
fimprog
```

```
int entrada(vet,t,c)
int vet[ ],t;string c;
{
  int L;
  imprima "\nEntrada do VETOR
  ", c "\n";
  para(L <- 0;L<t;L++)
  { imprima "Digite  numero  :
  ", L+1, ": "; leia vet[L];}
  retorna(vet);
}
```

```
int imprime(vet,t,c)
int vet[ ],t; string c;
{
  int L;
  imprima "\n VETOR ", c "\n";
  para(L <- 0;L<t;L++)
  { imprima "\n", L+1, " - ",
  vetL]; }
  retorna(vet);
}
```

```
void soma(vet,vet1,t)
int vet[ ],vet1[ ],t;
{
  int L,s;
  imprima "\nSOMA\n";
  para(L <- 0;L<t;L++)
  { s<-vet[L] + vet1[L];
  imprima "\n",s; }
}
```

```
void subtrai(vet,vet1,t )
int vet[ ],vet1[ ],t;
{
  int L,d;
  imprima "\nDIFERENCA\n";
  para(L <- 0;L<t;L++)
  { d<-vet[L] - vet1[L];
  imprima "\n",d;   }
}
```

Apêndice I

Interpretador UAL

O UAL (UNESA ALGORITHMIC LANGUAGE) é resultado do projeto final do curso de Informática da UNESA – Nova Friburgo de Andréa T. Medeiros e Adriana S. Spallanzani, sob a orientação do Prof. Juarez Muylaert Filho.

Começou a ser usado no segundo semestre de 2001 por todos os alunos do Curso de Informática da UNESA. Dessa forma, poderão surgir pequenos bugs, o que é normal.

Nesta parte, gostaríamos de reforçar alguns conceitos e dar algumas dicas para que você possa utilizar o **UAL** como uma ferramenta para seu aprendizado de algoritmos.

O **UAL** trabalha no ambiente Linux e é muito fácil de ser usado.

INSTALAÇÃO

- O usuário pode escolher os aplicativos que achar convenientes para utilização. O arquivo ual_v2.tar.gz contém o executável para interpretação simples.

- Faça o download do arquivo ual_V2.tar.gz. Após o download, o arquivo deve ser descompactado dentro do diretório **/usr/local.** Copie o arquivo para o diretório citado e descompacte-o, conforme os seguintes comandos:

1 – mude para o diretório usr/local, digitando: **cd /usr/local**
2 – copie para o diretório usr/local: **cp ual_V2.tar.gz /usr/local**
3 – digite: **tar zxvf ual_V2.tar.gz**

4 – digite: **mcedit /etc/profile** (pode ser outro editor)
5 – insira a linha: **PATH=$PATH:/usr/local/ual**
6 – reboot

Após esses procedimentos, o interpretador (em sua versão simples) já poderá ser utilizado.

Vamos dar algumas sugestões e tirar algumas dúvidas sobre este interpretador neste apêncie.

1. Na entrada de dados de uma variável do tipo **real**, sempre digite um ponto após o número se ele não tiver parte fracionária.

> Se você declarar uma variável como real, precisa digitar o *ponto* após o número caso ele não tenha parte fracionária.

Exemplo:

```
prog
 real a;
 imprima "\nnumero: ";
 leia a;
 imprima "\n",a;
 imprima "\n";
fimprog
```

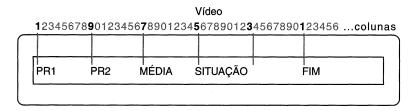
VÍDEO
root@localhost local]# digite nome do arquivo..ual
numero:12
Fail: Tipo incorreto na leitura da variável a.
root@localhost local]# digite nome do arquivo..ual
numero:12.
12.0

2. Você precisa converter para real um número inteiro para poder armazenar em uma variável declarada como real

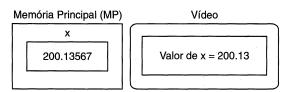
```
prog teste
 int a;
 real b;
 imprima "\nnumero: ";
 leia a;
 b <-intreal(a);
 imprima "\n",b;
 imprima "\n";
fimprog
```

VÍDEO
root@localhost local]# digite nome do arquivo..ual
numero:12
12.0
root@localhost local]#

3. Assim como em algumas linguagens, o UAL apresenta o caractere de controle **\t**:

Exemplo: **imprima** "PR1\tPR2\tMÉDIA\tSITUAÇÃO\tFIM";

Observe que a tela está "dividida" em áreas, contendo cada uma oito colunas. O símbolo **\t** levará a impressão para o início da próxima área desde que já não se encontre lá, como foi o caso do exemplo acima; o cursor, após a impressão da palavra SITUAÇÃO (por ter 8 letras), se posicionou na coluna 33, que é o início de uma nova área. Dessa forma, a palavra FIM foi deslocada para a coluna 41. É permitido o uso de vários **\t**.

4. O comando **formatar** é usado com o comando **imprima.**

Exemplo: x <- 200.13567;
imprima "Valor de x = ", **formatar**(x,2);

Este comando trunca a parte fracionária e converte o número real em string.

5. O UAL não permite a leitura de mais de uma variável num comando leia.

leia a, b ; *NÃO É PERMITIDO* Separe em dois comandos:
leia a ;
leia b ;

6. Observe que quando a parte inteira do número é PAR e a parte fracionária for 0.5, o número é truncado.

```
prog converte
 real b;
 imprima "\ndigite um numero real,
 colocando ponto";
 leia b;
 imprima "convertendo para inteiro:
 ",realint(b),"\n";
 imprima "\n";
fimprog
```

VÍDEO
[root@localhost local]# ual converte.ual *digite um numero real, colocando ponto: 12.5* *convertendo para inteiro: 12* *[root@localhost local]# ual converte.ual* *digite um numero real, colocando ponto: 12.51* *convertendo para inteiro: 13* *[root@localhost local]# ual converte.ual* *digite um numero real, colocando ponto: 13.5* *convertendo para inteiro: 14* *[root@localhost local]#*

Experimente acrescentar 0.01 ao valor de b -> realint(b + 0.01) e tudo ficará resolvido

7. Muitas soluções poderiam ter sido feitas com escolha mas preferimos fazê-las com **ses** aninhados para você poder testar no UAL.

8. Na versão 2 do UAL, não poderemos trabalhar com matrizes.

9. Por que escolhemos trabalhar com a sintaxe do UAL?

Reconhecemos que existem outros interpretadores conhecidos por todos nessa área, mas a impossibilidade de trabalhar com variáveis caracter e vetor caracter nos levou a escolher o UAL, pois sua sintaxe é muito parecida com a linguagem C, e como a maioria dos outros interpretadores mais completos está voltada para a linguagem PASCAL, proporcionamos a você uma outra possibilidade de aprender.

Apêndice II

Código ASCII

Código	Caractere	Código	Caractere	Código	Caractere	Código	Caractere	Código	Caractere
32		80	P	128	Ç	176	░	224	α
33	!	81	Q	129	ü	177	▒	225	ß
34	"	82	R	130	é	178	▓	226	Γ
35	#	83	S	131	â	179	│	227	π
36	$	84	T	132	ä	180	┤	228	Σ
37	%	85	U	133	à	181	╡	229	σ
38	&	86	V	134	å	182	╢	230	µ
39	'	87	W	135	ç	183	╖	231	τ
40	(	88	X	136	ê	184	╕	232	Φ
41	)	89	Y	137	ë	185	╣	233	Θ
42	*	90	Z	138	è	186	║	234	Ω
43	+	91	[	139	ï	187	╗	235	δ
44	,	92	\	140	î	188	╝	236	∞
45	-	93	]	141	ì	189	╜	237	φ
46	.	94	^	142	Ä	190	╛	238	ε
47	/	95	_	143	Å	191	┐	239	∩
48	0	96	`	144	É	192	└	240	≡
49	1	97	a	145	æ	193	┴	241	±
50	2	98	b	146	Æ	194	┬	242	≥
51	3	99	c	147	ô	195	├	243	≤
52	4	100	d	148	ö	196	─	244	⌠
53	5	101	e	149	ò	197	┼	245	⌡
54	6	102	f	150	û	198	╞	246	÷
55	7	103	g	151	ù	199	╟	247	≈
56	8	104	h	152	ÿ	200	╚	248	°
57	9	105	i	153	Ö	201	╔	249	∙
58	:	106	j	154	Ü	202	╩	250	·
59	;	107	k	155	ø	203	╦	251	√
60	<	108	l	156	£	204	╠	252	^{n}
61	=	109	m	157	Ø	205	═	253	^{2}
62	>	110	n	158	×	206	╬	254	■
63	?	111	o	159	ƒ	207	╧	255	
64	@	112	p	160	á	208	╨		
65	A	113	q	161	í	209	╤		
66	B	114	r	162	ó	210	╥		
67	C	115	s	163	ú	211	╙		
68	D	116	t	164	ñ	212	╘		
69	E	117	u	165	Ñ	213	╒		
70	F	118	v	166	ª	214	╓		
71	G	119	w	167	º	215	╫		
72	H	120	x	168	¿	216	╪		
73	I	121	y	169	⌐	217	┘		
74	J	122	z	170	¬	218	┌		
75	K	123	{	171	½	219	█		
76	L	124	|	172	¼	220	▄		
77	M	125	}	173	¡	221	▌		
78	N	126	~	174	«	222	▐		
79	O	127	•	175	»	223	▀		

Apêndice III

Raciocínio Lógico

As palavras "lógica" e "lógico" são familiares. Falamos freqüentemente de comportamento "lógico" em contraste a um comportamento "ilógico", de procedimento "lógico" em oposição a um "ilógico", de explicação "lógica" e de espírito "lógico". Em todos esses casos, a palavra "lógico" é usada, fundamentalmente, como "razoável".

Uma pessoa com espírito "lógico" é uma pessoa "razoável"; um procedimento "irrazoável" é aquele que se considera "ilógico". Todos esses usos podem ser considerados como derivativos de um sentido mais técnico dos termos "lógico" e "ilógico" para caracterizar os argumentos racionais.

O estudo da lógica é o estudo dos métodos e princípios usados para distinguir o raciocínio correto do incorreto. Naturalmente, essa definição não pretende afirmar que só é possível argumentar corretamente com uma pessoa que tenha estudado lógica. Afirmá-lo seria tão errôneo quanto pretender que só é possível correr bem, se estudou física e fisiologia necessárias para a descrição dessa atividade. Alguns excelentes atletas ignoram completamente os processos complexos que se desenrolam dentro deles próprios quando praticam o esporte.

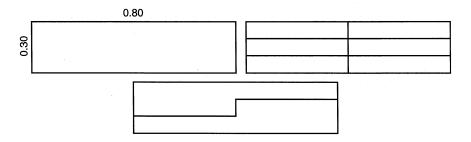
Agora, vamos praticar nossa lógica.

1. Um carpinteiro possui uma prancha de **0,80m** de comprimento e **0,30m** de largura. Quer cortá-la em dois pedaços **iguais** de modo a obter uma peça retangular que tenha **1,20m** de comprimento e **0,20m** de largura.

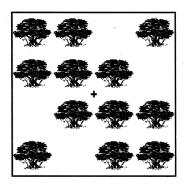
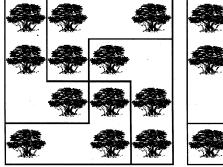
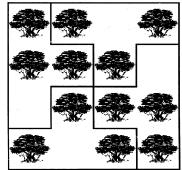
Divida a figura a seguir para atender às necessidades do carpinteiro.

0.80

0.30

Solução:

2. Um fazendeiro deixou, como herança para os seus quatro filhos, um terreno em forma de um quadrado no qual havia mandado plantar 12 árvores.

O terreno deveria ser dividido em quatro partes geometricamente iguais, contendo cada uma delas o mesmo número de árvores e todas, em alguma parte, tocando o centro do terreno.

Trace linhas na figura a seguir para atender às exigências do fazendeiro.

Soluções:

3. Certas propriedades relativas aos números inteiros recebem denominações curiosas que, não raras vezes, surpreendem os espíritos desprevenidos ou não-afeitos aos estudos das múltiplas transformações aritméticas.
220 e 284 são chamados de Números Amigos. Descubra por quê.

Solução:
Divisores de 220: 1 2 4 5 10 11 20 22 44 55 110 (não entra ele mesmo) -> somados -> 284
Divisores de 284: 1 2 4 71 e 142 (não entra ele mesmo) -> somados -> 220

4. Você saberia explicar por que a Matemática denomina *6, 28, 496 e 8128 de Números Perfeitos*?

Solução:
Divisores de 6: 1 2 3 (não entra ele mesmo) -> somados -> 6
Divisores de 28: 1 2 4 7 14 (não entra ele mesmo) -> somados ->28

5. Um mágico desafio sua platéia da seguinte maneira:

a. Pediu um número de cinco algarismos para alguém da platéia e este lhe deu: *37652*

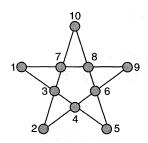
b. Comunicou à platéia que pediria um número, também de cinco algarismos, a outra pessoa e, imediatamente, escreveria um número (números em negrito), também de cinco algarismos embaixo do fornecido. Este passo seria repetido.

32456
67543
62134
37865

c. Garantiu à platéia que os cinco números somados dariam -> **237650**

Em que conceito da matemática ele se baseou para afirmar isso?

Solução:
Observe que cada número em negrito é Complemento a 9 do anterior; logo, a soma de 5 números daria 199998 e, por isso, você tira 2 do número dado pela primeira pessoa. Se fosse feito Complemento a dez, a soma daria 200000, o que lhe garante que o 2 seja colocado à frente do primeiro número.

6. Desejamos escrever os inteiros de 1 a 10 nas casas do desenho a seguir de tal forma que quaisquer quatro números alinhados apareçam em ordem crescente ou decrescente.

7. Com quatro, faço todos os números de 1 a 10.
Obs.: Use e abuse de operações, desde que não tenha mais do que 4 quatros e tenha, obrigatoriamente, 4 quatros.

$$1=\frac{44}{44} \qquad 2=\frac{4}{4}+\frac{4}{4} \qquad 3=\frac{4+4+4}{4} \qquad 4=\frac{4-4}{4}+4$$

$$5=\frac{4\times 4+4}{4} \qquad 6=\frac{4+4}{4}+4 \qquad 7=4+4-\frac{4}{4}$$

$$8=4+4+(4-4) \qquad 9=4+4+\frac{4}{4} \qquad 10=\frac{44-4}{4}$$

8. Mostre que os números inteiros de 1 a 16 podem ser dispostos em uma linha numa certa ordem, sem repetir nenhum, de modo que a soma de dois adjacentes (vizinhos) quaisquer seja um quadrado perfeito (isto é, o quadrado de um inteiro).

Solução: **8 1 15 10 6 3 13 12 4 5 11 14 2 7 9 16**
9 16 25 16 9 16 25 16 9 16 25 16 9 16 25

9. Monte uma expressão usando somente uma vez os algarismos de 1 a 9 de tal maneira que o resultado dê 100.

Solução: $1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 \times 9 = 100$

10. Como dispor oito oitos de forma que a soma seja 1.000?

Solução:
O número mais perto de 1.000 formado somente pelo algarismo 8 é 888.
Subtraindo 888 de 1.000, temos: 112.
O número mais perto de 112 formado somente pelo algarismo 8 é 88.
Subtraindo 88 de 112, temos: 24.
Como 24 é igual a 8 + 8 + 8, o problema ficará resolvido com a disposição feita a seguir:

$$888 + 88 + 8 + 8 + 8 = 1.000.$$

11. Complete as seqüências:

a) 1, 2, 6, 39, ___

b) 2, 10, 12, 16, 17, 18, 19,___

c) 3, 6, 10, 15, 21, 28, 36, 45, 55, __

Soluções:

a) $1525 = 39 * 39 + 4 = 1525$, $39 = 6 * 6 + 3$, $6 = 2 * 2 + 2$, $2 = 1 * 1 + 1$

b) a seqüência é composta pelos números que começam com a letra D, portanto o próximo nº é o 200.

c) esta seqüência é formada pela soma do primeiro número com si próprio que é o nº 3, e o resultado é somado com o nº utilizado para a soma na conta anterior, acrescido de 1.

Vejamos:

$3 + 3 = 6$

$6 + (3 + 1) = 10$

$10 + (3 + 1) + 1 = 15$

$15 + ((3 + 1) + 1) + 1 = 21$

Desta forma, podemos concluir que 45 somado com 10 é 55, logo o próximo número utilizado na soma será o 11, que somado com o 55, teremos o resultado 66.

12. Preencha o quadrado a seguir de tal maneira que a soma dos números que ficam sobre uma linha, ou sobre uma coluna, ou sobre uma diagonal, dê sempre 15 e todos os números têm de ser diferentes.

Solução:

2	9	4
7	5	3
6	1	8

13. A mãe de Takada tem cinco filhos: Tanaco, Taneco, Tanico, Tanoco. Qual é o quinto filho?

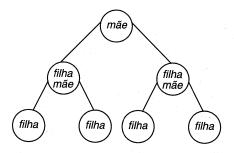
Solução:

Uma pessoa precipitada responde de imediato que o quinto filho é Tanuco. Evidentemente, a resposta correta é Takada, pois se a mãe de Takada tem cinco filhos é claro que Takada já é um dos filhos.

14. Depois que fiz uma conferência, perguntei ao organizador: " Quantas pessoas estavam presentes no auditório ? " Então, ele me respondeu: Se tivesse 50 pessoas a mais, a metade seria exatamente 500". Quantas pessoas assistiram realmente a conferência?

Solução:
Pela fala do organizador, podemos montar a seguinte expressão:
$(x + 50) / 2 = 500$,
logo $x + 50 = 500 \times 2$
$x + 50 = 1000$
$x = 950$

15. Entram num restaurante para jantar três mulheres, cada uma com duas filhas. Só existiam 7 lugares. Nenhuma ficou de pé. Como isso é possível?

Solução: Veja a figura a seguir:

16. Tenho 3 camisas: A, B e C. Uma é VERDE, uma é BRANCA e outra é AZUL, não necessariamente nessa ordem. Somente uma das afirmações a seguir é verdadeira: A é VERDE, B não é VERDE e C não é AZUL. Quais as cores de A, B e C, nessa ordem?

Solução:

A	B	C	1ª	2ª	3ª
azul	verde	branco	F	F	V
azul	branco	verde	F	V	V
branco	azul	verde	F	V	V
branco	verde	azul	F	F	F
verde	azul	branco	V	V	V
verde	branco	azul	V	V	F

17. Dentro de uma caixa fechada, há uma bola branca e uma bola preta. Numa segunda caixa fechada, há duas bolas brancas e, numa terceira caixa fechada, há duas bolas pretas. Cada caixa possui uma etiqueta indicando o conteúdo da caixa, mas alguém misturou as três etiquetas de modo que todas as etiquetas estão erradas. Você seria capaz de escolher apenas uma das seis bolas de modo tal que, olhando sua cor, você possa dizer o conteúdo de cada uma das caixas?

Possibilidades:

BP	BB	PP
PP	BP	BB
BB	PP	BP

Solução: Escolha a caixa com rótulo BP:

1. Se sair a bola PRETA, tudo resolvido, pois você tem certeza de que só existem dois grupos que têm pelo menos uma bola PRETA: BP e PP. Como BP não pode ser, logo só pode ser PP. Aí é só concluir que, na caixa BB, estarão as bolas BRANCA e PRETA e, na caixa PP, estarão as bolas BRANCA e BRANCA.
2. Se sair a bola BRANCA, tudo resolvido, pois você tem certeza de que só existem dois grupos que têm pelo menos uma bola BRANCA: BP e BB. Como BP não pode ser, logo só pode ser BB. Aí é só concluir que, na caixa BB, estarão as bolas PRETA e PRETA e, na caixa PP, estarão as bolas BRANCA e PRETA.

18. Um rei tinha cinco escravas. Duas tinham olhos pretos e sempre falavam a verdade. Três tinham olhos azuis e sempre mentiam. Um sábio matemático fez três perguntas cujas respostas foram:

a) a primeira escrava respondeu em um idioma que ele desconhecia quando ele lhe perguntou sobre a cor dos seus olhos.
b) a segunda escrava respondeu que a primeira tinha olhos azuis quando lhe foi perguntado que resposta a primeira tinha dado.
c) a terceira escrava respondeu que a primeira tinha olhos pretos e a segunda olhos azuis quando lhe foi perguntado qual a cor dos olhos das duas primeiras.

O sábio descobriu a cor dos olhos das escravas, na ordem em que se encontravam.

COMO O SÁBIO MATOU A CHARADA?

Solução:

a. A resposta da primeira escrava não interessava, pois o sábio tinha certeza da única resposta que poderia ser dada pela primeira: PRETO. Explicando: se ela falasse a verdade diria PRETO, mas se ela mentisse, ela também diria PRETO.

b. Ao perguntar para a segunda escrava o que a primeira tinha respondido, ela poderia definir a cor dos olhos da segunda tendo em vista a certeza da resposta da primeira. Quando ela falou que a primeira tinha olhos azuis, concluiu que a segunda escrava tinha olhos AZUIS.

c. Quando a terceira respondeu que a primeira tinha olhos PRETOS e a segunda tinha olhos AZUIS, concluiu que a terceira tinha olhos PRETOS, a segunda AZUIS, a primeira PRETOS e as outras duas AZUIS.

19. De três prisioneiros que estavam em cárcere, um tinha visão normal, o segundo era caolho e o terceiro era cego. Os três eram, pelo menos, de inteligência média. O carcereiro disse aos prisioneiros que, de um jogo de três chapéus brancos e dois vermelhos, escolheria três e os colocaria em suas cabeças. Cada um deles estava proibido de ver a cor do chapéu em sua própria cabeça. Reunindo-os, o carcereiro ofereceu a liberdade ao prisioneiro com visão normal se ele fosse capaz de dizer a cor do chapéu que tinha na cabeça. O prisioneiro confessou que não podia dizer e se retirou. A seguir, o carcereiro ofereceu a liberdade ao prisioneiro que só tinha um olho na condição de que ele dissesse a cor de seu chapéu. O caolho confessou que também não sabia dizê-lo e também se retirou. O carcereiro não se deu o trabalho de fazer a idêntica proposta ao prisioneiro cego, mas à insistência deste concordou-lhe em dar-lhe a mesma oportunidade. O prisioneiro cego abriu um amplo sorriso e disse: " Não necessito da minha vista; pelo que meus amigos disseram, vejo claramente *que o meu chapéu é* ____________________".

Solução:

a. O primeiro prisioneiro só poderia ter visto na cabeça dos outros dois as seguintes combinações de chapéus:

2º prisioneiro	3º prisioneiro
vermelho	vermelho
branco	branco
branco	vermelho
vermelho	branco

A primeira possibilidade está descartada, pois ele teria acertado, uma vez que só existiam dois chapéus vermelhos.

Sendo assim, restaram as três últimas.

b. O segundo prisioneiro, tendo em vista as possibilidades que restaram, só poderia ter visto:

3º prisioneiro
branco
vermelho
branco

Como a 1ª e a 3ª são iguais, ele só pode ter visto vermelho ou branco. Se ele tivesse visto vermelho, teria certeza de que seu chapéu era branco, pois se fosse vermelho, o primeiro teria acertado. Logo, ele só poderia ter visto chapéu branco no terceiro prisioneiro; daí sua dúvida: o dele seria branco ou vermelho?

c. Foram essas as conclusões que o terceiro tirou para dizer que o chapéu dele era *branco*.

20. Einstein escreveu este teste no século passado.
Ele afirmou que 98% do mundo não pode resolvê-lo... Você vai deixar ele tirar essa onda???

1. Há cinco casas de 5 diferentes cores.
2. Em cada casa mora uma pessoa de uma diferente nacionalidade.
3. Esses 5 proprietários bebem diferentes bebidas, fumam diferentes tipos de cigarro e têm um animal de estimação diferente.
4. Nenhum deles tem o mesmo animal, nem fuma o mesmo cigarro e nem bebe a mesma bebida.

A QUESTÃO É: QUEM TEM UM PEIXE?

Dicas:
O inglês vive na casa vermelha.
O sueco tem cachorros como animais de estimação.
O dinamarquês bebe chá.
A casa verde fica à esquerda da casa branca.
O dono da casa verde bebe café.
A pessoa que fuma Pall Mall cria pássaros.
O dono da casa amarela fuma Dunhill.

O homem que vive na casa do centro bebe leite.
O norueguês vive na primeira casa.
O homem que fuma Blends vive ao lado do que tem gatos.
O homem que cria cavalos vive ao lado do que fuma Dunhill.
O homem que fuma Bluemaster bebe cerveja.
O alemão fuma Prince.
O norueguês vive ao lado da casa azul.
O homem que fuma Blends é vizinho do que bebe água.

Solução:

AMARELA	AZUL	VERMELHA	BRANCA	VERDE
NORUEGUÊS	DINAMARQUÊS	INGLÊS	SUECO	ALEMÃO
ÁGUA	CHÁ	LEITE	CERVEJA	CAFÉ
DUNHILL	BLENDS	PALMALL	BLUEMASTER	PRINCE
GATOS	CAVALOS	PÁSSAROS	CACHORROS	PEIXE

A casa verde está à esquerda da branca. Lembre-se de que você estando de frente, ficará do lado direito.

Bibliografia

BEZERRA, R. Z., MACHADO, J. E. F. Mat2 – Matemática para o 2º grau.Rio de Janeiro: Ao Livro Técnico, 1979.

COPI, I.M. Introdução à lógica. trad. Álvaro Cabral. 2ª ed. São Paulo: Mestre Jou,. 1978.

FARRER, H., BECKER, C. G., FARIA, E. C. et al. Algoritmos estruturados. 2ª ed. Rio de Janeiro: LTC, 1998.

_____. Pascal Estruturado – Turbo Pascal. 3ª ed. Rio de janeiro: LTC, 1999.

GIOVANNI, R. J. , BONJORNO, J. R. Matemática – 2º grau. São Paulo: FTD, 1992.

GROSSI, E. Só ensina quem aprende. In: GROSSI, E. P., BORDIN, J. (org.). Paixão de aprender.5ª ed. Petrópolis: Vozes, 1992a. p. 69-75.

GUIMARÃES, A. M., LAJES, N. A. C. Algoritmos e estruturas de dados. Rio de Janeiro: LTC, 1985.

MANZANO, J. A. N. G., OLIVEIRA J. F. Algoritmos. Lógica para desenvolvimento de programação. São Paulo: Érica, 1996.

MEDEIROS, A.T., Spallanzani, A.S. Manual do UAL. Nova Friburgo: UNESA, 2000.

PINTO, Wilson Silva. Introdução ao desenvolvimento de algaritmos e estruturas de dados. 6ª ed. São Paulo: Érica, 1996.

Pontifícia Universidade Católica . Lista de exercícios da disciplina de Programação I. Curso de Informática. Rio de Janeiro: PUC-RJ,1999.

SALIBA, W. L. C. Técnicas de Programação – Uma Abordagem Estruturada. São Paulo: Makron Books, 1993.

SALVETTI, D. D., BARBOSA L. M. Algoritmos. São Paulo: Makron Books, 1998.

SCHILDT, H. C completo e total. 3ª ed. São Paulo: Makron Books, 1996.

SÉRATES, J. Raciocínio Lógico Vol. 1 e 2. editora: Jonofon Ltda. Brasília. 8ª ed. 1998.

SOUZA, J. C. M. O homem que calculava. Rio de Janeiro: Record. 2000.

_____. Matemática divertida e curiosa. 15ª ed. Rio de Janeiro: Record. 2001.

STEINBRUCH, A. Matrizes, determinantes e sistemas de equações lineares. São Paulo: McGraw-HILL, 1989.

TREMBLAY, J.P., BUNT, R.B. Ciência dos computadores: uma abordagem algorítmica. São Paulo: Mc Graw Hill do Brasil, 1989.

UCCI, W., SOUZA, R. L., KOTANI, A. M. Lógica de programaoção: os primeiros passos. 5ª ed. São Paulo: Érica, 1995.

Universidade Estácio de Sá. Lista de exercícios da disciplina de Programação I. Curso de Informática. Rio de Janeiro: UNESA-Bispo,2001.

Universidade Estácio de Sá. Lista de exercícios da disciplina de Programação I. Curso de Informática. Rio de Janeiro: UNESA-Nova América,2001.

VILLAS, M.V., VILLASBOAS, L. F. P. Programação: conceitos, técnicas e linguagens. Rio de Janeiro: Campus, 1997.

WIRTH, N. Algorithms + data structure = programs. Englewood Cliffs, N. J. , Prentice-Hall, 1976.

_____. Programação sistemática. Rio de Janeiro: Campus, 1978.